INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in "sectioning" the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again – beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

University Microfilms International
300 North Zeeb Road
Ann Arbor, Michigan 48106 USA
St. John's Road, Tyler's Green
High Wycombe, Bucks, England HP10 8HR
JOHWANI, NAARESH DHARAN
STUDY OF A CLASS OF PARALLEL PROGRAMES.
RICE UNIVERSITY, PH.D., 1978
RICE UNIVERSITY

STUDY OF A CLASS OF PARALLEL PROGRAMS

by

N. D. JOTWANI

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

THESIS DIRECTOR'S SIGNATURE:

[Signature]

HOUSTON, TEXAS

AUGUST, 1977
Acknowledgements

The author records with pleasure his sincere gratitude to Dr. J. Robert Jump for his guidance and advice. When the research seemed to have entered a dead-end, he gave the author a generous second opportunity to bring it to an acceptable conclusion.

Sincere thanks are also due to Drs. Blattner, Hirschberg, and Kennedy for their interest in the project, and to A. R. Hajare and S. M. Mayekar for companionship.

The research was supported by NSF under grant GJ-750.
Appendix I 124
Appendix II 127
Appendix III 131
References 134
CHAPTER 1

INTRODUCTION AND BASIC DEFINITIONS

1.1 Introduction

A. The main aim of this research is an investigation into the implications of using top-down design (i.e. design by successive refinement steps) for parallel programs. To this end, we use a model for parallel programs based on Free Choice Petri nets (FCP nets), which is obtained by associating operators and predicates with vertices of the net.

Based on an analysis into the structural properties of FCP nets, we develop an algorithm which finds a top-down program simulating a given parallel program. We thus prove the intuitive result that any parallel program can be simulated by a top-down parallel program. However, we find that in some cases the degree of parallelism attainable in the top-down program is necessarily smaller than that in the original, where the measure of parallelism is the number of different ways of carrying out a computation. We investigate fully this phenomenon of loss of parallelism, determining necessary and sufficient conditions for a given parallel program to be simulated by a top-down parallel program without loss of parallelism.

The present model may be viewed as a natural extension of the familiar flow-chart notation to parallel programs. FCP nets can represent in a natural way programs written, for example, in a FORK - JOIN type of language. The present study can therefore be viewed as an investigation into the properties of well-formed (in a sense to be defined precisely) parallel flow-charts.
B. Various studies in the past have been directed at investigating different properties of parallel systems, using algebraic and/or graphical system models. Properties of parallel systems such as determinacy, freedom from deadlocks, equivalence, and others, have been investigated, and in most cases necessary and sufficient conditions have been found under which the respective properties hold. Another related area has been schemes for coordination of sequential processes. Synchronization primitives of greater or lesser power have been proposed, studied, and compared.

At a more theoretical level, Petri nets have been suggested as the basis on which parallel systems can be modeled. Though many of the decidability questions related to Petri nets have been answered, liveness and safeness conditions for the general class of Petri nets have not been determined. Restriction classes of Petri nets have been studied successfully; marked graphs and FCP nets fall into this category.

Another theoretical approach has been that of parallel program schemata. Keller has presented results concerning the determinacy of a parallel program schema, and the conditions under which a schema is 'maximally parallel'. Intuitively, a parallel program schema is maximally parallel if the only constraints placed on the operations are those necessitated by data-dependencies within the operator-set. In a sense, therefore, the analysis of (13), and that in this dissertation, are both concerned with the degree of parallelism attainable in parallel systems, though the two approaches are considerably different.
The 'interpretation' we use in this dissertation is similar to that in (13), though an explicit representation of program variables is not made here. The present model may be viewed as a finite-state realization of a parallel program schema, and may be compared with the realization in (13), which is based on state-transition graphs.

The advantages of top-down design are widely recognized. Program design, validation, as well as optimization, data-flow analysis, etc., are greatly facilitated when top-down design methods are employed. In the case of parallel programs, especially, the complexity of these procedures increases enormously unless the programs are designed by step-wise refinement. In view of the results in this dissertation concerning loss of parallelism, therefore, we have discovered a performance factor which may offset some of the advantages of top-down design.

C. Remaining Sections of this Chapter present basic definitions related to Petri nets, and the essential properties of FCP nets. Chapter 2 contains formal definitions of the model, its 'behaviour', and the 'simulation' of one parallel program by another. Chapter 3 contains definitions related to top-down design of parallel programs, and outlines the similarities between these definitions and other well-known concepts in the area of sequential program graphs. Chapter 4 is an analysis into the structure of well-formed FCP nets, based on which Chapter 5 presents a Structure Algorithm, which finds a top-down program simulating a given parallel program. Chapter 6 presents conditions for a parallel program to be simulated by a top-down program without loss of parallelism. Chapter 7, finally, summarises the work done with a few concluding remarks.
1.2 Petri Nets

A Petri net is a directed, bipartite graph in which the two types of vertices are usually called places and transitions. Graphically, the two types of vertices are represented as shown in the following simple net:

Here $t_1, t_2, t_3$ are transitions, and $p_1, p_2$ are places. Associated with a Petri net is a marking, which assigns to each place a non-negative number of tokens or markers. The marking on a net determines which transitions may be fired (if any). It may be thought of as the enabling rule for the 'operations' which correspond to the transitions. In the above net, for example, $t_1$ is firable, and on the firing of $t_1$, one of the two tokens on $p_1$ will be transferred to $p_2$.

In the following definitions we shall formalize the ideas outlined above.

**Def.1.2.1** A finite Petri net $G$ is a triple $G=(T,P,E)$ where:
- $T$ is a finite non-empty set of transitions,
- $P$ is a finite non-empty set of places,
and $E \subseteq (T \times P) \cup (P \times T)$ is the set of edges defining a directed bipartite graph.

A marking $M$ on Petri net $G=(T,P,E)$ is a function $M: P \rightarrow \mathbb{N}$, where $\mathbb{N}$ is the set of non-negative integers.

In the following pages, we shall tacitly assume 'net' to mean 'finite net'. We shall denote the sets $\{ y \mid (x,y) \in E \}$ and $\{ y \mid (y,x) \in E \}$ by $x'$ and '$x$ respectively. Also, for any $X \subseteq T \cup P$,
we denote by $X'$ and $\cdot X$ respectively the sets \( \{ y \mid y \in x \} \) for some \( x \in X \) and \( \{ y \mid y \in \cdot x \} \) for some \( x \in X \).

**Def.1.2.2** A firing function associated with a transition $t$ is defined iff $\forall p \in \cdot t$, $M(p) > 0$. This function on the set of all markings of net $G$ is then given by the following, where $M' = f(M, t)$ is the value of the function at $M$:

$$M'(p) = \begin{cases} 
M(p) + 1, & \text{if } p \in \cdot t - t' \\
M(p) - 1, & \text{if } p \in t' - t \\
M(p), & \text{otherwise.}
\end{cases}$$

We denote by $\delta$ the union of all the firing functions associated with the transitions of the net $G$, and we say $M' = \delta(M)$ if $M' = f(M, t)$ for some transition $t$ in $G$. We say a transition $t$ is firable at marking $M$ if its associated firing function is defined at $M$. A firing sequence $\sigma$ of net $G$ is a string $\sigma = t_1t_2 \ldots t_m$, $t_i \in T$ for $1 \leq i \leq m$, which satisfies the property that for a sequence $M^0, M^1, \ldots, M^m$ of markings over $G$, $M^i = f(M^{i-1}, t_i)$ for $1 \leq i \leq m$. $M^0$ is the initial marking on $G$. We extend the notation here to say that $M^m = \delta(M^0, \sigma)$.

We shall use the operator '.' to denote concatenation of strings representing firing sequences of $G$.

**Def.1.2.3** A transition $t$ in $G$ is live iff, for any firing sequence $\sigma$ of $G$, another sequence $\sigma_t$ can be found s.t. $t$ is firable under the marking $\delta(M^0, \sigma, \sigma_t)$. The net $G$ is live iff all its transitions are live.

A place $p$ in $G$ is safe iff, for any firing sequence $\sigma$ of $G$, $M'(p) \leq 1$ where $M' = \delta(M^0, \sigma)$. The net $G$ is safe iff all its places are safe.

This completes the definitions related to Petri nets.
1.3 Basic Definitions

We shall define in this Section some basic ideas which will be used frequently in the following pages, as well as some notation which will prove useful later.

**Def.1.3.1** A subnet (or subgraph) of net $G=(T,P,E)$ is a net $G'=(T',P',E')$ s.t. $T' \subseteq T$, $P' \subseteq P$ and $E'$ is the restriction of $E$ to $T'$ and $P'$. We shall say a subnet is t-complete iff for every transition $t$ in the subnet, the set $'t \cup t'$ is contained in the set of places of the subnet.

We shall denote by $T^{-1}(G')$ and $P^{-1}(G')$ respectively the sets of transitions and places of a subnet $G'$ of net $G$.

**Def.1.3.2** An (elementary) path of length $m$ in net $G$ is a sequence of vertices $u_0 u_1 \ldots u_m$ s.t. $(u_i, u_{i+1}) \in E$ for $0 \leq i \leq m-1$, and no $u_i$ appears more than once in the sequence. We denote the path as $\Pi(u_0, u_m)$, and we shall also speak of it as a subnet of the net $G$. We shall say that $u_0$ and $u_m$ are the initial and terminal end-points, respectively, of $\Pi$. Vertices other than the end-points will be referred to as the internal vertices of $\Pi$. Two paths are said to be i-disjoint if they have only 1 or 2 end-points (and no other vertices) in common.

We shall refer to an initial subpath or a terminal subpath of a path $\Pi$ with obvious connotations. We shall use ' . ' to denote the operation of concatenation of paths (recall '.' is also used to denote concatenation of firing sequences). A cycle of length $m$ in $G$ is a sequence $u_0 u_1 \ldots u_m$ of vertices of $G$ which satisfies : $(u_i, u_{i+1}) \in E$ for $0 \leq i \leq m-1$, $(u_m, u_0) \in E$, and no $u_i$ appears more than once in the sequence.
1.4 Free Choice Petri Nets

The class of Petri nets is considerably more general than the class we have chosen for our model in this paper. We are interested in live and safe nets for our model of parallel programs and, consequently, we shall restrict our attention to a suitable sub-class of Petri nets for which the conditions of liveness and safeness have been well-understood. Free Choice Petri nets, we claim, provide a natural choice for parallel programs. After defining this class, we shall show an example and illustrate the naturalness of the resulting representation.

**Def.1.4.1** A Petri net \( G = (T, P, E) \) is **Free Choice** iff

\[ \forall p \in P, |p'| > 1 \Rightarrow '(p') = \{p\}. \]

The implication of the above definition is that whenever a place is marked which has more than one output transitions, any one of these transitions may fire independently of the state of the other places in the net. Also, note that the above definition may be restated as

\[ \forall t_1, t_2 \in T, t_1 \cap t_2 \neq \emptyset \Rightarrow |t_1| = |t_2| = 1. \]

**Ex.1.4.1** The following net is Free Choice, because \( p \) is the only place in it with more than one output transitions, and \( \{p\} = '(p') \).

![Diagram](image)

We know that conditional branches in programs have the general semantic structure: if \( (\text{condition} = \text{TRUE}) \) then goto label.
In other words, the branch is taken independently of the state of the (parallel) program at all other points (i.e. the values of all other program counters). We claim that a natural representation of parallel programs results if we restrict the decision nodes in it to have this Free Choice property. The two way branch is then represented in our model as shown below

\[ \text{where } r_1 = r_2 = \{p\}. \]

The token on p may be utilised to fire \( t_1 \) or \( t_2 \) independently of the marking over the rest of the net, just as the conditional branch above may be taken one way or the other independently of the rest of the parallel program.

The Free Choice property is not satisfied by the simple Petri net below. In this net, the choice made at \( p_1 \) determines the direction in which the branch will be taken at \( p_2 \), which does not have a 'free choice'.

We shall exclude from our program model branches such as the one at \( p_2 \) above. The places such as \( p_1 \) above, i.e. those having more than one outputs, we shall call the Free Choice places (FC places). We denote by FC(G) the set of Free Choice places of Free Choice Petri net (FCP net) G.

FCP nets do not provide us with a representation for synchronization schemes in our programs, since these, in general, require non-Free Choice places. However, two points may be noted in this regard:

i. For many purposes of analyses, such as determinacy, or
restricted forms of 'well-formedness', we do not want to include the synchronization schemes in our model. In such cases we think of the synchronization schemes as distinct from the program and superimposed on it, so that they may be omitted when not required for the analysis.

ii. Mutual exclusion, a very basic type of synchronization, has been incorporated into our model in a perfectly natural way without destroying its Free Choice property. This is achieved by assigning the same 'operator' (or 'predicate') to more than one vertices of the FCP net. The mutual exclusion present at the corresponding vertices is then implicit rather than explicit. A similar approach may be seen in the models of (10,11).

The class of FCP nets was analyzed by Hack(6), who found necessary and sufficient conditions for the liveness and safeness of FCP nets. We shall use Hack's result as our starting point for investigating structural properties of our model.

Two sub-classes of the class of FCP nets will be of special interest to us in the following pages.

Def.1.4.2 A Petri net G=(T,P,E) is said to be a state machine if each t ∈ T has at most one input place and at most one output place, i.e. |t| ≤ 1 and |t'| ≤ 1. G is said to be a marked graph if each p ∈ P has at most one input transition and at most one output transition, i.e. |p| ≤ 1 and |p'| ≤ 1.

A state machine corresponds to the familiar sequential finite state system, i.e. one having no parallelism. Clearly a state machine is live and safe if it is strongly connected and has exactly one token.

A marked graph represents, in a sense, the simplest possible
parallel system, one in which there are no decision nodes. The properties of marked graphs have been determined successfully and marked graphs have been employed in building parallel system models.

State machines and marked graphs isolate two orthogonal properties of parallel systems: (i) branches/merges in flow of control, and (ii) initiation/termination of parallel sequences. The class of FCP nets is a natural combination of these two smaller classes. This fact will be brought out in a very elegant fashion below when we describe the two dual reduction procedures in terms of which Hack found the liveness and safeness of FCP nets.

**Def.1.4.3** An FCP net is said to be well-formed iff there exists a marking $M$ on it s.t. the net is live and safe under $M$.

**Ex.1.4.2** Consider the following FCP net $G$ which is under a live and safe marking:

Now consider the following two-step reduction on $G$:

i. delete the edge $(p,t_2)$

ii. find the largest strongly connected component of the resulting net.

This reduction will yield the following net, which is seen as just that subgraph of $G$ which will be visited an unbounded number of times if we always fire $t_1$ (and therefore never $t_2$)
when the place \( p \) is marked. As we shall see below, this subgraph is obtained through the reduction described in Def. 1.4.5.

The criterion of well-formedness given below is simply a formal and complete statement of the principle illustrated in the above example.

**Def. 1.4.4** Let \( G = (T, P, E) \) be an FCP net. An MG-\textit{allocation} AM on \( G \) is a function \( AM: \text{FC}(G) \rightarrow T \) satisfying \( \forall p \in \text{FC}(G), AM(p) \in p' \).

**Def. 1.4.5** Let \( G = (T, P, E) \) be an FCP net and let AM be an MG-allocation on \( G \). The set of strongly connected marked graph components (SCMG components) of \( G \) corresponding to AM is the set of mutually disjoint subnets of \( G \) found through the reduction procedure below:

i. delete all the edges of \( G \) in the set \( FC(G) \times [FC(G)' - AM(FC(G))] \), i.e. the 'unallocated output edges' of FC places,

ii. find the maximal strongly connected components of the resulting graph, and

iii. of the components found in ii, delete those which are not t-complete; the remaining components are the SCMG components of \( G \) corresponding to allocation AM.

**Note** The reduction described above is a slight re-statement of Hack's original formulation, but the two can easily be shown to be equivalent. We call the above reduction the MG-reduction on \( G \) with allocation AM. We now state Hack's criterion of well-formedness in terms of the above reduction.

**Thm. 1.4.1** FCP net \( G = (T, P, E) \) is well-formed iff, for any MG-allocation AM on \( G \), the following conditions are satisfied:
(a) the set of SCMG components of $G$ corresponding to $AM$ is not null, and
(b) each SCMG component is a marked graph.

SCMG components have a natural, intuitive meaning associated with them, as brought out by Ex.1.4.2. Each SCMG component $R$ consists of just those vertices of $G$ which will be reached if the transitions of $G$ are fired under the fixed decision pattern at the PC places corresponding to an MG-allocation which yields $R$. In view of this we shall denote as a Trace of $G$ any SCMG component $R$ of $G$.

The second reduction procedure, described below, is the dual of the first, but is more general in the following sense. It provides us also with a characterization of live and safe markings on a well-formed FCP net, in addition to the criterion of well-formedness itself.

**Def.1.4.6** Let $G=(T,P,E)$ be an FCP net. An SM-allocation $AS$ on $G$ is a function $AS:T \to P$ satisfying $\forall t \in T, AS(t) \in t$. **

**Def.1.4.7** Let $G=(T,P,E)$ be an FCP net and let $AS$ be an SM-allocation on $G$. The set of strongly connected state machine components (SCSM components) of $G$ corresponding to $AS$ is the set of mutually disjoint subnets of $G$ found through the reduction procedure below:

i. delete all the edges of $G$ in the set $[P-AS(T)] \times T$, i.e. the 'unallocated input edges' of transitions:

ii. find the maximal strongly connected components of the resulting graph, and

iii. of the components found in ii, delete those which are
not p-complete **. The remaining components are the SCSM components of G corresponding to allocation AS.

We call the above reduction SM-reduction with allocation AS. Before presenting the criterion of well-formedness in terms of the above reduction, we present two useful properties of SCSM components.

**Lemma 1.4.1** Let G be a well-formed FCP net under live marking M. Let S be an SCSM component of G with m tokens under M, and let M' be any other marking of G reachable from M through some firing sequence σ. Then there are exactly m tokens on S under M'.

**Lemma 1.4.2** Let well-formed FCP net G be under marking M. Let S be an SCSM component of G which is blank (i.e. has no tokens on it) under M. Then no transition on S can be made firable in G.

Both these Lemmas are proved in (6), in a slightly different form. We shall denote as a **1-SCSM component** an SCSM component of well-formed FCP net G which has exactly one token on it. A parallel program may be thought of as a number of sequential programs which have been synchronized at various points. (This synchronization may be in the form of JOIN statements, for example.) In the context of well-formed FCP nets, each 1-SCSM component represents one such parallel program.

**Thm. 1.4.2** FCP net G=(T,P,E) under marking M is live and safe iff, for any SM-allocation AS on G, the following are satisfied:

(a) the set of SCSM components of G corresponding to AS

**A subnet G' of G is p-complete iff for every place p on G', the transitions 'p∪p' are on G'.**
is not null,

(b) each SCSM component is a state machine with at least one token on it under marking M, and

(c) each place p in G is on a 1-SCSM component of G.  

Thus the two reduction procedures yield components which isolate, respectively, the marked graphs and the state machines which make up a well-formed FCP net.

Further inferences can be made regarding the structure of well-formed FCP nets, using Thms.1.4.1-1.4.2 as starting points and applying certain elementary properties of finite strongly connected directed graphs. In this Section, we present a few of these properties, while in Chapter 4 we shall deduce from these some useful structural features of FCP nets. The proofs of the following Lemmas are given in Appendix I.

**Lemma 1.4.3** Let G be a well-formed FCP net. Let S' (resp. R') be a strongly connected subgraph of G obtained at step ii of SM-reduction (resp. MG-reduction). Then S' (resp. R') is a subgraph of an SCSM component S (resp. SCM component R) of G.  

**Lemma 1.4.4** Let Π be an elementary cycle in well-formed FCP net G and let x, y be distinct vertices on Π. Let τ(x, y) be a path in G s.t. x and y are the only vertices of τ on Π. Then if x is a transition, y is a transition.

**Lemma 1.4.5** Let R=(T_R, P_R, E_R) be a strongly connected subgraph of well-formed FCP net G s.t. R is a marked graph satisfying \( \forall t \in T_R, t' \subseteq P_R \). Then R is \( t \)-complete, i.e. \( \forall t \in T_R, t' \subseteq P_R \).  

**Lemma 1.4.6** Let \( \Pi, \tau, x \) and y in well-formed FCP net G be as in Lemma 1.4.4 above. Let \( \Pi = \Pi_1(x, y) . \Pi_2(y, x) \) s.t. \( \Pi_1 \) and \( \Pi_2 \)
are i-disjoint subpaths of $\Pi$. Further, let $\Pi, \tau$ be such that there is no path in $G$ from an internal vertex of $\tau$ to an internal vertex of $\Pi_1$ which does not contain $x$. Then if $x$ is a place, $y$ is a place.

This completes our description of FCP nets, and the criteria for their well-formedness, liveness, and safeness. Our parallel program model based on FCP nets is defined in the next Chapter. In investigating the properties of our model, we shall further analyze the structural features of well-formed FCP nets.
CHAPTER 2

DEFINITION OF THE MODEL

In this Chapter we formally define our parallel program model, designated here by the name 'formal parallel program'. We also define the concepts of the behaviour of a parallel program, and the simulation of one parallel program by another.

The notions of the behaviour of a parallel program, and the simulation between parallel programs, are based on the set of 'behaviour sequences' of a program, defined in Section 2.2. These sequences are obtained from the firing sequences of the Free Choice Petri net so as to record the order in which the various operations and decisions are carried out. The behaviour sequences defined here are similar to the computation sequences which characterize other parallel program models.\(^{(12,13)}\) The last Section of this Chapter introduces the notion of 'valid transformations' on parallel programs — we shall see in latter Chapters that some of the descriptions and proofs of the algorithms presented there are simplified when we make use of this notion.

2.1 Parallel Program Model

In order to base our parallel program model on an FCP net, we shall modify a live and safe FCP net and draw it in the following form:

![FCP net diagram]

'start' place

'end' place
In the above diagram, it is implicit that (a) \( t^0_0 \) is the only transition firable under the initial marking, and (b) addition of the 'return link', as shown below, to the above graph yields a strongly connected live and safe FCP net as characterized in the previous Chapter.

By this means we obtain from a live and safe FCP net a parallel program flow-graph with unique initial and terminal end-points ('start' and 'end' places above). The FCP net shown in the first diagram above will be given the name linear FCP net in the following pages.

Using the linear FCP net as our starting point, we now define our model for parallel programs as follows:

**Def. 2.1.1** A formal parallel program (FPP) is a 5-tuple \( \mathbf{F} = (G, S_{op}, f_{op}, S_{pr}, f_{pr}) \) where

i. \( G = (T, P, E) \) is a linear FCP net under initial marking \( M^0 \),

ii. \( S_{op} \) is a set of operators and \( f_{op} \) is a total function \( f_{op} : T \to S_{op} \cup \{ \lambda \} \), and \( \lambda \) is the null-operator, \( \lambda \notin f_{op}[FC(G)] \),

iii. \( S_{pr} \) is a set of predicates and \( f_{pr} \) is a total function \( f_{pr} : FC(G) \to S_{pr} \).

It is possible to extend this definition so that every transition is associated with a set \( r(t) \) of range cells and a set \( d(t) \) of domain cells, and every FC place is associated with a set \( d(p) \) of domain cells. Then \( r \) and \( d \) would be functions
with range $\mathbb{M}$, where $\mathbb{M}$ would be the set of memory locations (or variables) of the program. However, the analysis of this paper does not require an explicit definition of program variables, which has therefore been excluded from this paper.

**Def. 2.1.2** Let $\mathcal{F} = (G, S_{\text{op}}, f_{\text{op}}, S_{\text{pr}}, f_{\text{pr}})$ be an FPP. We say that the 4-tuple $(S_{\text{op}}, f_{\text{op}}, S_{\text{pr}}, f_{\text{pr}})$ is an interpretation on $G$ which yields $\overline{f}$.

Note that the above definition of interpretation is similar to that in (13). The operator set and interpretation of (13) are analogous, respectively, to the sets $S_{\text{op}}, S_{\text{pr}}$ and the functions $f_{\text{op}}, f_{\text{pr}}$ of our model. With the exception of program variables $r(\cdot)$ and $d(\cdot)$, therefore, an FPP may be thought of as a realization of a parallel program schema.

To draw a further parallel with the model of (13), note that an FPP is necessarily a finite-state realization of a compact, commutative schema. The present form of realizations is then seen to correspond more naturally with flow-chart notation extended to parallel programs than the form of realizations which makes use of state-transition graphs to represent parallel systems.

Comparing our model with that of (19), we see that we have a much smaller number of vertex types and, consequently, simpler enabling rules. Also, the model of (19) is of a 'data-flow' type, i.e. the function nodes receive data at the input edges and the results of a computation are placed on the output edges.

Points of similarity and dissimilarity may also be found with various other models of parallel computation which have been proposed.\(^{(2)}\)
Recall that a Trace (i.e. an SCMG-component) of a live and safe FCP net is a strongly connected marked graph, and that transitions on this marked graph may be fired indefinitely if the appropriate branches are taken at the FC places. In a linear FCP net, however, there are two types of Traces: those which contain 'start', and those which do not. Only the second type of Traces in a linear FCP net are strongly connected marked graphs, and they represent potential iterations in the corresponding FPP. Traces which contain 'start' are not strongly connected marked graphs, because the return link from 'end' to 'start' is not present in a linear FCP net.

**Ex.2.1.1** Consider the following graph:

This graph has one Trace which contains 'start', and one which does not. The two Traces are drawn separately below:

Those Traces of a linear FCP net which contain 'start' are not potential iterations in the corresponding FPP; these we shall denote as the **basic** Traces of a linear FCP net. A linear FCP net in which every Trace is a basic Trace will be designated
as a basic FCP net. Intuitively, such a net corresponds to a parallel program without iterations.

One feature of parallel programs which is brought to light when we make use of linear FCP nets as described above is illustrated below:

**Ex.2.1.2** Consider the following net:

![Diagram of a net](image)

This graph has one potential iteration (i.e. Trace not containing 'start') which is shown in bold lines. The number of times this iteration is carried out is exactly the number of times $t_1$ fires, and equals the number of times $t_5$ fires. However, the $k^{th}$ firing of $t_5$ may come after the $(k+1)^{th}$ firing of $t_1$, for $k \geq 1$, because of the presence of the initial token at $q$. ##

This type of parallelism associated with initial tokens (other than the token on 'start') is given here the name pipeline parallelism in view of its resemblance to a marked graph representation of a conventional pipe-line as shown below:

![Diagram of a marked graph](image)

In the following two Sections of this Chapter we shall define the notions of the behaviour of an FPP, and the simulation of an FPP by another. In making these definitions, as well as in the latter Chapters, we shall make use of a convention
for naming the vertices of a linear FCP net \( G \) which will simplify the resulting notation. In the remaining paragraphs of this Section we describe this naming convention. Let \( G = (T, P, E) \).

**i.** Elements of \( S_{op} \) are named \( f_0, f_1, \ldots, f_{|S_{op}|-1} \). Elements of \( T \) are then named according to the scheme below:

- a. \( \{\lambda^1, \lambda^2, \ldots, \lambda^n\} = f_{op}^{-1}(\lambda), \) i.e. \( m_\lambda = |f_{op}^{-1}(\lambda)| \),
- b. \( t_0 \) is the single transition at the output of 'start',
- c. \( \{t_1^0, t_1^1, \ldots, t_i^m\} = f_{op}^{-1}(f_i), \) i.e. \( m_i = |f_{op}^{-1}(f_i)| \), for \( 0 \leq i \leq |S_{op}|-1 \).

We say that there are \( m_{i+1} \) occurrences of the operator \( f_i \) (or the \( i \)th operator) in \( \mathcal{F} \), for \( 0 \leq i \leq |S_{op}|-1 \).

**ii.** Elements of \( S_{pr} \) are named \( g_0, g_1, \ldots, g_{|S_{pr}|-1} \). Elements of \( FC(G) \) are then named so that \( p_1^0, p_1^1, \ldots, p_i^m \) are all the elements of \( f_{pr}^{-1}(g_i) \), for \( 0 \leq i \leq |S_{pr}|-1 \). We say that there are \( n_{i+1} \) occurrences of the predicate \( g_i \) (or the \( i \)th predicate) in \( \mathcal{F} \), for \( 0 \leq g_i \leq |S_{pr}|-1 \).

The naming convention is chosen so as to provide us with an easy translation from an FPP to the corresponding FCP net and vice versa. In the following pages we shall often consider programs for which \( f_{op} \) and \( f_{pr} \) must be assumed to be one-to-one for the sake of generality. We shall use \( f_i, g_j \) etc. in such cases as vertex names in the net \( G \), and we shall refer to a transition as an operator, or to an FC place as a predicate.

To carry out the identification of linear FCP nets with FPPs a step further, we shall say "FCP net \( G \) has property \( x \)" to mean that any interpretation on \( G \) will yield an FPP with that property. This completes our model definition, and we now present the other necessary definitions related to it.
2.2 Behaviour of a Parallel Program

The behaviour of an FPP, the parallel program model of the previous Section, will now be defined along the lines of the 'computation sequences' of other models (12, 13). Specifically, the set of all possible sequences of operators and predicates that may be observed will define the behaviour of an FPP.

**Def.2.2.1** Let \( F = (G, S_{op}, f_{op}, S_{pr}, f_{pr}) \) be an FPP. Let \( \alpha \) be any firing sequence of \( G \). The behaviour sequence \( \alpha' \) of \( F \) corresponding to \( \alpha \) is obtained from \( \alpha \) by means of the following sequence of steps:

i. delete any instances of \( \lambda^i_i, 1 \leq i \leq m_\lambda \), from \( \alpha \),

ii. insert \( p^i_j \) immediately to the left of any instances of an \( x \in (p^i_j)' \) in the resulting string, where \( 0 \leq i \leq n_j \) and \( 0 \leq j \leq |S_{pr}| - 1 \),

iii. replace any instances of \( t^i_j \) in the resulting string by the operator \( f_j \), where \( 0 \leq i \leq m_j \) and \( 0 \leq j \leq |S_{op}| - 1 \),

iv. replace any instances of \( p^i_j \) in the resulting string by the predicate \( g_j \), where \( 0 \leq i \leq n_j \) and \( 0 \leq j \leq |S_{pr}| - 1 \).

**Ex.2.2.1** Consider the FPP shown below:

![Diagram of an FPP](image)

Two of the firing sequences of the above net are \( t^0_0 t^1_0 t^0_2 \) and \( t^0_0 t^1_1 t^0_2 \). The corresponding two behaviour sequences are \( f^0_0 g^0_0 f^0_0 f^2_2 \) and \( f^0_0 g^0_1 f^1_1 f^2_2 \). We see that a behaviour sequence preserves the information about the decisions made at FC places, as well as the order in which the various operations and decisions were carried out.
We denote by $\beta(F)$ the set of all behaviour sequences of $F$. We then denote by $\beta$ the onto function $\beta : F(G) \to \beta(F)$ which is defined by steps i - iv of Def. 2.2.1.

Def. 2.2.2 The behaviour of an FPP $F$ is the set $\beta(F)$ of all the behaviour sequences of $F$.

Based on the above definition of the behaviour of an FPP, in the following Section we define the simulation of an FPP $F$ by another FPP $F'$.

2.3 Simulation

Using the above definition of the behaviour of an FPP, we now define the idea of simulation between two FPPs.

Recall that the presence of parallelism, in general, permits a computation to be carried out in more than one way. The following definition states the conditions under which two behaviour sequences of an FPP represent the same computation.

Def. 2.3.1 Let $\alpha$ and $\alpha'$ be any two behaviour sequences of an FPP $F = (G, S_{op}, f_{op}, S_{pr}, f_{pr})$. $\alpha$ and $\alpha'$ are said to be similar if the following conditions are satisfied:

i. $\#(f_i | \alpha) = \#(f_i | \alpha')$, $\forall f_i \in S_{op}$

$\#(g_i | \alpha) = \#(g_i | \alpha')$, $\forall g_i \in S_{pr}$

ii. if the $k^{th}$ occurrences of any $g_i \in S_{pr}$ are followed in $\alpha$ and $\alpha'$ by $f_1$ and $f_1'$ respectively, then $l = l'$; here $g_i \in S_{pr}$, $f_1, f_1' \in S_{op}$ and $1 \leq k \leq \#(g_i | \alpha) = \#(g_i | \alpha')$.

The second part of the definition above states that the pattern of decisions made to obtain the two sequences $\alpha$ and $\alpha'$ is identical. We say $(\alpha, \alpha') \in \mathit{sim}$ iff $\alpha$ and $\alpha'$ are similar.

** Here $\#(x | \alpha)$ denotes the number of occurrences of $x$ in the string $\alpha$. $F(G)$ is the set of all firing sequences of $G$.


Clearly $\sim$ is then an equivalence relation on $\mathcal{B}(\mathcal{F})$.

**Def.2.3.2** Let $\mathcal{F}$ and $\mathcal{F}'$ be two FPPs. We say $\mathcal{F}'$ simulates $\mathcal{F}$ iff the following conditions are satisfied:

i. $\mathcal{B}(\mathcal{F}') \subseteq \mathcal{B}(\mathcal{F})$,

ii. no equivalence class of the relation $\sim$ is disjoint with $\mathcal{B}(\mathcal{F}')$.

We say that $\mathcal{F}$ and $\mathcal{F}'$ are equivalent, or $\mathcal{F}'$ simulates $\mathcal{F}$ without loss of parallelism, if $\mathcal{B}(\mathcal{F}) = \mathcal{B}(\mathcal{F}')$.

We show below an example illustrating the above definitions.

**Ex.2.3.1** Consider the following three FPPs:

\[
\begin{align*}
\mathcal{F}_1' & \quad \mathcal{F}_2' & \quad \mathcal{F}_3'
\end{align*}
\]

It can easily be verified that $\mathcal{F}_3'$ simulates $\mathcal{F}_1'$ but is not equivalent to it, because $f_4$ has been placed in sequence with $g_0,f_1,f_2,f_3$. $\mathcal{F}_2'$ does not simulate $\mathcal{F}_1'$ because ii of Def.2.3.2 is violated. There is no behaviour sequence in $\mathcal{F}_2'$ similar to any of the sequences $f_0g_0f_3', f_0f_4g_0f_3'$, $f_0g_0f_3f_4, f_0g_0f_3f_4f_5$ and $f_0f_4g_0f_3f_5$ of $\mathcal{F}_1'$.

This completes the definitions related to our model which are essential for the analysis in the latter Chapters.
2.4 Valid Transformations

Before we describe some of the algorithms in this paper, we shall introduce the notion of a valid transformation on a linear FCP net $G$. A transformation $T$ on $G$ is valid iff it satisfies the following two conditions:

(a) the output net is a linear FCP net,
and (b) for any given interpretation on the original net, an interpretation can be found on the output net s.t. the output FPP simulates the original.

Usually a transformation will consist of sequences of steps of two basic types:

(i) We introduce 'null' transitions into the net. In this case the output interpretation is found simply by assigning the $\lambda$-operator to the newly introduced transitions.

(ii) We introduce into the net a subgraph $G''$ isomorphic to some original subgraph $G'$ of the net. In this case the output interpretation is found by assigning to vertices of $G''$ the same operators, predicates, etc. as those of the corresponding vertices in $G'$.

For most of the transformations in this paper, we shall be able to prove in a relatively straight-forward way that, provided the output interpretations are found as described, the transformations are valid. We shall therefore omit the equations describing the interpretations for the output nets in terms of those of the original nets.

Also, to simplify the algebra, we shall denote by $G$ both the input and the output nets. We shall therefore describe a transformation on $G$ by saying that $G$ satisfies some structural
property 'X' after the transformation, say $T$, is applied to it. This is true, for example, of transformations $T_1 - T_3$ of Section 4.2, i.e. each of these is shown simply to ensure that a certain property holds in $G$.

To further simplify the notation, we may say of a transformation that the output FCP net simulates the original if condition (b) above of valid transformations is satisfied for an appropriate interpretation on the output net.

The conventions described in this Section will enable us to give a concise description of the various transformations on FPPs which are presented in the following Chapters.
CHAPTER 3

TOP-DOWN DESIGN

The major portion of this thesis is concerned with the implications of using top-down design techniques, i.e. design by successive refinement steps, for parallel programs. We shall define in this Chapter three different classes of FPPs, which are obtained when we permit different types of restricted control structures to be introduced into the program in one step of refinement. The three classes are: top-down programs (TDPs), interval reducible programs (IRPs), and structured programs (SPs). We shall see that the class of IRPs is a proper subclass of the class of TDPs, and that the class of SPs is a proper subclass of the class of IRPs.

In Chapter 6 we shall show that under certain conditions an FPP has no TDP equivalent, even though any FPP can be simulated by a TDP. In the present Chapter we shall outline why, from the point of view of this loss of parallelism (and under a definition of equivalence which refers to the functional behaviour of parallel programs) the three classes may be thought of as equivalent. In other words, under the modified definition of equivalence, any TDP has an SP equivalent.

**Top-down Programs**

**Def. 3.1** A proper marked graph (pmg) is a linear FCP net which is a marked graph, and from which the places 'start' and 'end' (along with the edges incident on them) have been deleted.

Deletion of the two places and the incident edges is as diagrammed below:
Initial and terminal transitions in the pmg are as shown.

**Def. 3.2** A proper state machine (psm) is a linear FCP net which is a state machine, and from which the place 'start' (along with its output edge) has been deleted and a terminal transition added as diagrammed below:

The above two restricted types of FCP nets are the only permissible control structures that may be introduced in one step of the process of successive refinement which defines top-down programs, as brought out in the following definitions.

**Def. 3.3** The substitution of net $G'$ into a linear FCP net $G$ is defined iff $G'$ is either a pmg or a psm. The substitution at transition $t$ of $G$ consists in replacing $t$ in $G$ by $G'$ as shown below, to yield another linear FCP net $G''$. 
We shall use the terms S-substitution or M-substitution, respectively, to denote that the net \( G' \) is a pmg or a psm in a particular instance of substitution.

**Def. 3.4** The class of top-down FCP nets, which is strictly contained in the class of linear FCP nets, is defined inductively as follows:

**basis step:** The net \( G_0^0 \) shown below is a top-down FCP net

\[
\begin{array}{c}
\text{\textit{start}} \quad \circ \quad \text{(G}_0^0 \quad \text{(end}}
\end{array}
\]

**induction step:** If linear FCP net \( G \) is a top-down net, and if net \( G'' \) is obtained from \( G \) by means of a single substitution step, then \( G'' \) is a top-down FCP net.

The psm or pmg nets which are substituted at various steps in obtaining a top-down FCP net \( G'' \) will be referred to as the constituent psm or pmg nets of \( G'' \), respectively.

**Def. 3.5** An FPP \( F' = (G, S_{op}', f_{op}', S_{pr}', f_{pr}') \) is a top-down program (TDP) iff \( G \) is a top-down FCP net.

Essentially, the definition of TDPs states that modules representing parallelism, and those representing control flow logic, i.e. pmg and psm nets respectively, should be introduced separately into the program, one at a time. The next two classes of FPPs, the classes IRP and SP, are then arrived at by adding further restrictions to the above definition of TDPs. More specifically, the next two classes are obtained by further restricting the state machines that may be introduced into an FCP net during an S-substitution. These restrictions on state machines are frequently used on sequential program flowgraphs.\(^{(8,15)}\)
Interval Reducible Programs

The class of interval reducible sequential flow-graphs was introduced so as to provide a theoretically sound definition of 'reducibility', which would be useful in global flow analysis, optimization, etc. The class IRP of parallel programs is now obtained by restricting the state machines introduced through S-substitutions to be interval reducible.

We shall give below a constructive definition of interval reducible state machines, show that it is equivalent to the standard definition, and finally, that any TDP has an IRP equivalent.

Def. 3.6 A 1/n reducible state machine (rsm) is a state machine obtained through one or more applications of the following steps:

i. basis step: The net shown below is a 1/1 rsm

\[ \xrightarrow{\text{basis step}} \]

ii. composition: Two 1/1 rsm nets \( G' \) and \( G'' \) may be composed as shown below to yield a 1/1 rsm \( G \)

\[ G' \xrightarrow{\text{composition}} G'' \]

iii. \((k_1, k_2)\) merge/branch: Given a 1/m rsm \( G' \), with \( m > k_1 > 1 \) and \( k_2 \geq 1 \), a 1/(m-\( k_1 + k_2 \)) rsm \( G \) is obtained through the addition of a place \( q \) with \( k_1 \) input transitions and \( k_2 \) output transitions, as shown below

\[ G' \xrightarrow{\text{merge/branch}} G \]
iv. iteration: Given a 1/m rsm $G'$, for $m \geq 2$, a 1/(m-1) rsm $G$ is obtained by adding to $G'$ a place $q$, a transition $t$, and three edges, as shown below.

![Diagram]

We now show that any 1/m rsm is an interval reducible graph; we shall need a few basic definitions for this, which are given below. The usual definitions have been modified slightly to apply them to bi-partite state machines.

Let $G=(T,P,E)$ be a state machine with a distinguished transition $t^0$ satisfying the condition that there is a path in $G$ from $t^0$ to every other vertex. We say a vertex $x$ in $G$ dominates vertex $y$ iff every path in $G$ from $t^0$ to $y$ contains $x$.

An interval $G'=(T',P',E')$ in $G$ with header $h$ is a strongly connected subgraph of $G$ satisfying (i) $h$ dominates every vertex in $G'$, and (ii) every cycle in $G'$ contains $h$. The input edges of $h$ in $G'$ are said to be the back-edges of $G'$. (Note that since $G$ is a state machine, $h$ must be a place.) We say $G'$ is collapsed to a single place $p'$ when $G'$ is replaced in $G$ by $p'$ as shown in the following diagram.

![Diagram]

To show that any 1/m rsm is interval reducible, let $G$ be a 1/m rsm, and let \{q_1, ..., q_n\} be the set of places of $G$ introduced during iteration steps (see Def. 3.6). Let $G^i$ be the largest
strongly connected component of $G$ s.t. $q_1$ dominates every vertex in $G^i$. Then there exists at least one $G^i$ s.t. $q_1$ is the only element of $\{q_1, \ldots, q_n\}$ on $G^i$. But then $G^i$ defines an interval of $G$, and may be collapsed to a single place $p^i$. Also, any acyclic $1/m$ rsm is interval reducible, since there is a path in it from the initial transition to any other vertex. Therefore induction on the number of times iteration is used shows $G$ is an interval reducible state machine.

Conversely, let $G$ be any interval reducible state machine. Let $G'$ be one of its intervals, and let $G'^a$ be the acyclic graph obtained on deleting the back-edges of $G'$. By induction on the number of edges in $G'^a$, it can easily be shown that it is a $1/m$ rsm for some $m \geq 1$. Now every back-edge in $G'$ can be re-introduced into it by means of a single application of the iteration rule. Therefore $G'$, along with its input and output transitions, is a $1/n$ rsm, for some $n \geq 1$. Induction on the number of intervals in $G$ yields that it is an rsm.

Therefore we have established that $G$ is an interval reducible state machine iff it is a $1/n$ rsm for some $n \geq 1$. In view of this observation, we now define the class IRP.

**Def.3.7** A top-down FCP net $G$ is said to be interval reducible iff every instance of $S$-substitutions employed in obtaining $G$ introduces a $1/1$ rsm into it.

**Def.3.8** An FPP $\mathcal{F} = (G, S_{op}, f_{op}, S_{pr}, f_{pr})$ is an interval reducible program (IRP) iff $G$ is interval reducible.

It can easily be shown that any TDP has an IRP equivalent, since it is well known that a state machine may be replaced by an equivalent interval reducible state machine.
Structured Programs

We now define the class $\mathbf{SP}$ using a definition similar to the one in (18). Once again, the definition is in terms of a restriction on permissible $S$-substitutions.

Def. 9 A structured state machine (ssm) is a psm obtained through one or more applications of the following rules

i. basis step the following nets are ssm nets

\[
\begin{array}{c}
\text{Net 1} \\
\text{Net 2} \\
\text{Net 3}
\end{array}
\]

ii. induction step if $G$ and $G'$ are ssm, and $G''$ is obtained by substituting $G'$ in $G$ at transition $t$ of $G$ (see Def. 3.3), then $G''$ is an ssm.

Def. 3.10 A top-down FCP net $G$ is said to be structured iff every instance of $S$-substitutions employed in obtaining $G$ introduces an ssm into it.

Def. 3.11 An FPP $\mathcal{F} = (G, S_{op}, f_{op}, S_{pr}, f_{pr})$ is a structured program (SP) iff $G$ is structured.

As shown, for example in (18), every sequential program has an equivalent structured program, where equivalence is defined in terms of the functional behaviour of programs. It is shown in (18) that the above result holds only if extra variables are introduced into the program which serve as flags. In effect, the flags are used to transform every multiple-exit loop into a single-exit loop.

Our definition of simulation and equivalence, however, rules out the introduction of extra variables, and operations on them, into the programs. If however, we re-define equivalence in terms of the functional behaviour of programs (and incorporate
the notion of augmenting the variable set of a program) then we can show, using techniques similar to those in (18), that any TDP has an SP equivalent. As in the case of IRPs, this is because, under the appropriate definition of equivalence, any psm can be transformed into an equivalent ssm, without any loss in the degree of parallelism attainable.

A full development of these ideas relating the three classes of programs defined in this Chapter is outside the scope of the present work. A related presentation may be seen in (13), where Keller has related two definitions of equivalence (based on 'computation sequences' and 'program history', respectively) on the class of parallel program schemata.

In Chapter 6 we shall present necessary and sufficient conditions under which an FPP has a TDP equivalent. In view of the above remarks concerning IRP and SP classes (keeping in mind the two different criteria of equivalence mentioned) it follows that the same conditions will apply to classes IRP and SP as well.
CHAPTER 4

ANALYSIS OF FCP NETS

In this Chapter we shall analyze linear FCP nets further, in terms of the two reduction procedures of Chapter 1. The analysis is similar to the analysis of sequential program graphs into intervals, strongly-connected regions, etc. In a way, the present analysis can be thought of as reducing the 'global' reduction criteria of Chapter 1 into 'local' structural properties of linear FCP nets. In the following Chapter we shall make use of the present analysis to devise an algorithm which finds a top-down FPP simulating any given FPP.

4.1 Structure of a Trace

We saw in Chapter 1 that a well-formed FCP net can be decomposed into strongly-connected components through one of two dual reduction procedures. The components obtained through MG-reduction are named Traces because they correspond to the paths the tokens will trace in the net if transitions are fired indefinitely long with a fixed pattern of decisions at FC places. In linear FCP nets, only those Traces which do not contain 'start' can be thought to correspond to loops in sequential programs. Each Trace of a linear FCP net which does not contain 'start' is a strongly-connected marked graph and a t-complete subgraph of the net.

We now examine some of the properties of the Traces of linear FCP nets. Consider the following Trace $R = (T_R, P_R, E_R)$ of some net $G$.
Notation Let $C(R, p_1)$ denote the set
\[ \{ \tau \mid \tau \text{ is an elementary cycle in } R \text{ s.t. } p_1 \text{ is incident on } \tau \} \]

We claim that $C(R, p_1)$ represents, in a natural sense, the 'level of parallelism' of the place $p_1$ in marked graph $R$. To see this, note that $R$ has two elementary cycles
\[ \tau_1 = p_3 \cdot t_1 \cdot p_1 \cdot t_2 \]
and \[ \tau_2 = p_3 \cdot t_1 \cdot p_2 \cdot t_2 \]
and that therefore
\[ C(R, p_1) = \{ \tau_1 \} \]
\[ C(R, p_2) = \{ \tau_2 \} \]
and \[ C(R, p_3) = \{ \tau_1, \tau_2 \} . \]

Since, intuitively, $p_3$ is at a higher level of parallelism in the above graph than $p_1$ and $p_2$, we see that $C(R, p_1)$ can indeed be said to reflect the level of parallelism of $p_1$ in $R$. In other words, we have
\[ C(R, p_1) = C(R, p_3) \implies p_1 \text{ and } p_3 \text{ are at the same level of parallelism in } R \]
\[ C(R, p_1) \subset C(R, p_2) \implies p_1 \text{ is at a higher level of parallelism than } p_1 \text{ in } R \]

To formalize the above notions we define an equivalence relation $\sigma_1$ and a partial order $\sigma_2$ on $P_R$, the set of places of a Trace $R$:

i. $\forall p_i, p_j \in P_R, (p_i, p_j) \in \sigma_1 \iff C(R, p_i) = C(R, p_j)$
i. $\forall p_i, p_j \in P_R, (p_i, p_j) \in \sigma_2 \iff p_i = p_j \text{ or } C(R, p_j) \subset C(R, p_1)$
For the simple Trace \( R \) of the previous diagram we then have \( \sigma_1 = \emptyset \) while \( \sigma_2 \) is given by the Hasse diagram below:

\[
\begin{array}{c}
\circ P_3 \\
| \\
\circ P_1 \downarrow \\
\circ P_2
\end{array}
\]

**Ex. 4.1.1** Consider the following Trace in some net 3

\[
\begin{array}{c}
\circ P_1 \\
\circ P_2 \leftarrow \circ P_3 \leftarrow \circ P_4 \\
\circ P_5 \leftarrow \circ P_9 \leftarrow \circ P_8 \\
\circ P_7 \leftarrow \circ P_6
\end{array}
\]

Here \( \sigma_1 \) will be given by the directed graph below:

\[
\begin{array}{c}
\circ P_1 \leftarrow \circ P_2 \leftarrow \circ P_3 \leftarrow \circ P_4 \leftarrow \circ P_5 \\
\circ P_6 \leftarrow \circ P_7 \leftarrow \circ P_9 \leftarrow \circ P_8
\end{array}
\]

\( \sigma_2 \) is given by the following Hasse diagram:

\[
\begin{array}{c}
\circ P_1 \leftarrow \circ P_2 \\
\circ P_3 \leftarrow \circ P_4 \leftarrow \circ P_5 \\
\circ P_7 \leftarrow \circ P_8 \leftarrow \circ P_9 \leftarrow \circ P_6 \\
\circ P_2 \leftarrow \circ P_3 \leftarrow \circ P_4 \leftarrow \circ P_5 \leftarrow \circ P_6
\end{array}
\]

We now describe a basic property of a Trace in an FCP net which follows immediately from the above definitions. We shall see later that this property is used in further analyzing the net. The property is the first of the various 'local' properties of linear FCP nets which we shall describe.

**Lemma 4.1.1** Let \( R = (T_R, P_R, E_R) \) be a Trace in linear FCP net \( G \) and let \( p_1, p_2 \in P_R \) s.t. there exists a path \( \tau(p_1, p_2) \) in \( G \) and no internal vertices of \( \tau \) are on \( R \). Then \( C(R, p_2) \subseteq C(R, p_1) \).
Proof Assume the opposite, i.e. let $\pi \not\in C(R, p_2) - C(R, p_1)$. Since $R$ is strongly-connected we can then find an elementary path $\alpha(t, p_1)$ from a transition $t$ on $\pi$ to $p_1$. This gives us the following configuration in $G$:

Then by considering a suitable MG-allocation of $G$ consistent with the above subgraph, we can find an MG-component $R'$ of $G$ s.t. the above subgraph is wholly contained in $R'$. But the in-degree of $p_2$ in the above subgraph is greater than 1, which contradicts the fact that each MG-component of a linear FCP net is a marked graph. Our assumption must therefore be wrong and the result follows.

In the next Section we examine the structure of a linear FCP net in terms of a partial order on the set of all Traces of the net. In Section 4.3 we shall examine some further properties of a Trace in a linear FCP net.

4.2 Structure of a Linear FCP Net

We now analyze the structure of a linear FCP net in terms of a partial order on the set of its Traces, i.e. on $\mathcal{R}(G)$. The partial order will be seen to correspond to the intuitive notion of 'one iteration being present in one of the parallel branches of another'. As a simple example of this relationship between two Traces, consider the following net.
Ex. 4.2.1 Let $G$ be the linear FCP net below:

The net has three Traces, $R_1, R_2$ and $R_3$ as shown. It is seen that $R_2$ is an iteration within exactly one of the parallel branches of $R_1$; similarly $R_3$ is an iteration within one of the two parallel branches of $R_2$. We shall see that this relation can be defined in a formal way for a linear FCP net.

As a starting point for the definition of this relation between the Traces of a linear FCP net, consider the following definition:

Def. 4.2.1 Let $p \in P$ in a linear FCP net $G=(T,P,E)$. The subgraph of $G$ in the neighbourhood of $p$, denoted by $SN(p)$, is the subgraph of $G$ obtained through the following steps:

1. Let $P^{-1}[SN(p)] = \{ p \}$ and $T^{-1}[SN(p)] = \emptyset$
2. For any place $q$ in $P^{-1}[SN(p)]$, add all its output transitions, i.e. all the transitions in $q'$, to $T^{-1}[SN(p)]$
3. For any transition $t$ in $T^{-1}[SN(p)] \cup U$, if all the blank places in $t'$ are in $P^{-1}[SN(p)]$, then add all the input and output places of $t$, i.e. all the places in $t \cup t'$, to $P^{-1}[SN(p)]$
4. Repeat steps 1 and 2 till no more places or transitions can be added to $SN(p)$

** Under the initial marking $M^0$
step 5. Let $U = \{ t \in T^{-1}[SN(p)] \mid t' \in P^{-1}[SN(p)] \land t \notin P^{-1}[SN(p)] \}$

step 6. If $U \neq \emptyset$ then go to step 1

step 7. $E^{-1}[SN(p)]$ is the restriction of $E$ to $P^{-1}[SN(p)]$ and $T^{-1}[SN(p)]$

We show below an example of the above definition. We shall see later in this Section that this definition is central to the partial order $\sigma_{SN}$ which we shall describe on $(R(G),)$ the set of all Traces of $G$.

Ex. 4.2.2 Consider the net of the previous example. For places $p_i$, $1 \leq i \leq 4$, the subgraphs $SN(p_i)$ are shown below.

$SN(p_1) = SN(p_2)$:

$SN(p_3)$:

$SN(p_4)$:

It can easily be shown that for any linear net $G$, $SN('start') = G$. In a certain sense, $SN(p)$ defines the region of $G$ which is reachable from $p$. We know that in a parallel system, one process can be thought of as initiating one or more other processes. Here $SN(p)$ contains exactly those vertices of $G$ which the 'process' $p$ can initiate.

Note Consider the case when 'end' $\in P^{-1}[SN(p)]$ for some $p \in P-\{ 'start' \}$. 
For the sake of uniformity of presentation in some of the transformations, we shall think of the net $G$ as being augmented by the vertices and edges shown in bold lines below:

\[ \text{'start'} \xrightarrow{t_0} \xrightarrow{FCP \text{ net}} \text{'end'} \xrightarrow{\lambda_e} \]

This device will permit us a more concise description of $T_2, T_3$ later in this Section, and of the algorithms in Chapter 5, than would be possible otherwise.

**Def. 4.2.2** Let $SN(p)$ be as defined above for a linear net $G$. The set of **last transitions** of $SN(p)$, denoted $T_{\text{last}}(p)$, is the set

\[
T_{\text{last}}(p) = \{ t \in T^{-1}[SN(p)] \mid t' \notin P^{-1}[SN(p)] \}
\]

The set $T_{\text{last}}(p)$ is, intuitively, the set of end transitions of $SN(p)$, as may be seen from the values of $T_{\text{last}}(p)$ given below for the previous example.

**Ex. 4.2.3** For the net $G$ of the previous two examples, the sets $T_{\text{last}}(p_i)$, $1 \leq i \leq 4$, are as follows:

\[
T_{\text{last}}(p_1) = T_{\text{last}}(p_2) = \{ t_1 \}
\]
\[
T_{\text{last}}(p_3) = \{ t_3 \}
\]
\[
T_{\text{last}}(p_4) = \{ t_2 \}
\]

These may be compared with the diagrams of the previous example.

Also note that if we consider the net $G$ augmented by the vertices and edges as described above, then for any $p \in P-\{ \text{'start'} \}$, 'end' $\in P^{-1}[SN(p)] \Rightarrow \lambda_e \in T_{\text{last}}(p)$.

We are now ready to define the partial order $\sigma_{SN}$ on the
set $\mathcal{R}(G)$, as mentioned at the beginning of this Section. It is given as follows:

$$\forall R_i, R_j \in \mathcal{R}(G),$$

$$(R_i, R_j) \in \sigma_{SN} \iff R_i = R_j \text{ or}$$

- there exists an FC place $p$ on $R_i$ s.t. (a) $C(R_i, p)$ does not contain all the blank cycles of $R_i$ **, and
- (b) $R_j$ is a subgraph of $SN(p)$

$\sigma_{SN}$ can easily be shown to be reflexive, transitive and anti-symmetric. The precise meaning of the condition (a) on the right hand side above will be clear after the definitions of Section 4.3. At this point, it may be taken as the formal equivalent of the intuitive statement: $p$ is on one (of the more than one) parallel sequence in $R_i$.

This formulation then shows that a linear FCP net imposes an inherent, natural, partial order on the set of its Traces.

**Ex. 4.2.4** For the net of the previous examples, $\sigma_{SN}$ is given by the Hasse diagram below:

```
\begin{center}
\begin{tikzpicture}
  \node (R1) at (0,0) {$R_1$};
  \node (R2) at (0,-1) {$R_2$};
  \node (R3) at (0,-2) {$R_3$};
  \node (R4) at (0,-3) {$R_4$};
  \draw[->] (R1) -- (R2);
  \draw[->] (R2) -- (R3);
  \draw[->] (R3) -- (R4);
\end{tikzpicture}
\end{center}
```

**Ex. 4.2.5** Consider the net $G$ with its Traces $R_1, \ldots, R_4$ shown below

```
\begin{center}
\begin{tikzpicture}
  \node (R1) at (0,0) {$R_1$};
  \node (R2) at (0,-1) {$R_2$};
  \node (R3) at (0,-2) {$R_3$};
  \node (R4) at (0,-3) {$R_4$};
  \node (start) at (-1,0) {'\textit{start}'};
  \node (end) at (1,0) {'\textit{end}'};
  \draw[->] (start) -- (R1);
  \draw[->] (R1) -- (R2);
  \draw[->] (R2) -- (R3);
  \draw[->] (R3) -- (R4);
  \draw[->] (R4) -- (end);
\end{tikzpicture}
\end{center}
```

** For a basic Trace of $G$, replace 'blank cycle' by 'blank path from $t_0$ to 'end'. ('Blank' refers to the initial marking.)
The partial order \( \sigma_{SN} \) for this net is given by the Hasse diagram below.

Clearly this corresponds to the fact that \( R_2 \) is an iteration within one of the parallel sequences of \( R_1 \), etc. Any Trace \( R \) of a linear FCP net \( G \) which contains 'start' will clearly be a maximal element of the set \( \mathcal{R}(G) \) under partial order \( \sigma_{SN} \).

We now present some of the properties of a linear FCP net in terms of the above definitions.

**Lemma 4.2.1** Let \( R=(T_R,P_R,E_R) \) be a Trace in linear FCP net \( G \) and let \( p \in FC(G) \cap P_R \) s.t. \( C(R,p) \) does not contain all the blank cycles of \( R \). Then any blank cycle wholly contained in \( SN(p) \) is not wholly contained in \( R \). Further, such a cycle of \( G \) is wholly contained in a Trace \( R' \) of \( G \) s.t. \( (R,R') \in \sigma_{SN} \).

**Proof** Assume the opposite, i.e. that a cycle \( \tau \) as described is wholly contained in \( R \). From the definition of \( SN(p) \), and in view of the premise that there is another blank cycle \( \tau' \) in \( R \) s.t. \( p \) is not on \( \tau' \), it follows that the paths shown below must exist in \( R \).

```
```

But the in-degree of \( q \) above is 2, which contradicts the fact that \( R \) is a marked graph. Our assumption is wrong, i.e. \( \tau \) is

**See foot-note on p. 42 above.**
not wholly contained in $R$.

Further, by considering an MG-allocation of $G$ consistent with the cycle $\tau$, we can find a Trace $R'$ of $G$ wholly contained in $\text{SN}(p)$. In view of the definition of $\text{SN}(p)$, such an allocation can always be found. This yields $(R, R') \in \sigma_{\text{SN}}$ and the proof is complete.

We shall see in the following Chapter that the definition of $\text{SN}(p)$ is central to the algorithms described therein. In particular, the observations we make in the remaining part of this Section, pertaining to some of the properties of $\text{SN}(p)$, will be used more than once in that Chapter.

Lemmas 4.2.2 - 4.2.4 below will be used to prove the validity of transformations $T_1 - T_3$ which will follow.

**Lemma 4.2.2** Let $p$ be an arbitrary place in linear FCP net $G=(T, P, E)$. If an SCSM-component $S$ of $G$ contains $p$ then it contains exactly one element of $T_{\text{last}}(p)$.

**Proof** From the definition of $\text{SN}(p)$, we know that if $S$ contains $p$ then it contains at least one element of $T_{\text{last}}(p)$. To show that $S$ contains exactly one element of $T_{\text{last}}(p)$, assume the opposite; i.e. let $t_1, t_2 \in T_{\text{last}}(p) \cap P^{-1}(S)$. Then the following paths are present in $S$ and in $\text{SN}(p)$, where $r \notin P^{-1}[\text{SN}(p)]$:

Now consider any Trace $R$ of $G$ containing $p, q, t_1$ and $r$, and consider the place $s$ on $R$ (and $S$) where a continuation of the path $\tau_3(q, t_2)$ meets $R$ (see diagram below).
Then there is a cycle $\Pi$ in $R$ which contains $r$ and $s$, but not $p$ and $q$; i.e. $C(R,s) \not\subseteq C(R,q)$. In view of the path $\tau_3', \tau'_4$ above, however, we have a contradiction to Lemma 4.1.1. Our assumption must be wrong, and the result follows. ##

**Lemma 4.2.3** Let $p$ be any place in linear net $G$. If an SCSM-component $S$ of $G$ contains one element of $T_{\text{last}}(p)$ then it also contains $p$. ##

**Lemma 4.2.4** Let $p$ be any place in linear net $G$. For any $t \in T_{\text{last}}(p)$, there exists an SCSM-component $S$ of $G$ s.t. $S$ contains $t$ and $p$. ##

The proofs of the above two Lemmas follow arguments similar to those of Lemma 4.2.2. The transformations described below, which are used more than once in the next Chapter, apply to the subgraph $SN(p)$ in the case when $SN(p)$ does not wholly contain a Trace $R$ of $G$. The transformations ensure that $SN(p)$ satisfies certain properties necessary for the algorithms of the next Chapter.

1. Let transformation $T_1$ be as follows:

   **$T_1$ input:**
   
   i. linear FCP net $G=(T,P,E)$
   
   ii. place $p \in P$ satisfying the condition that $SN(p)$ does not wholly contain a Trace $R$ of $G$

   **output:** net $G$ modified so that
   
   i. $\forall p \in P^{-1}[SN(p)] \cap T_{\text{last}}(p)$, $M^0(p) = 0$
ii. $\forall t \in T_{\text{last}}(p), t \cdot \cap P^{-1}[\text{SN}(p)] = \emptyset$

and

iii. the transformation is valid

step 1. Repeat the following till i. above is satisfied:

Let $x$ be any marked place in $T_{\text{last}}(p) \cap P^{-1}[\text{SN}(p)]$.

Then $x$ is as shown on the left below. Replace $x$ by a blank place $y$ and the additional vertices and edges as shown on the right below.

$$
\begin{array}{c}
t' \quad x \\
\uparrow \\
t' \quad y
\end{array}
\quad \Rightarrow 
\begin{array}{c}
t' \\
\uparrow \\
\lambda_t
\end{array}
\quad 
\begin{array}{c}
t' \\
\uparrow \\
y \\
\lambda_t \\
t
\end{array}
$$

here $z \notin P^{-1}[\text{SN}(p)]$

$x \in P^{-1}[\text{SN}(p)]$

and $t \in T_{\text{last}}(p)$

step 2. Repeat the following till ii. above is satisfied:

Let $t$ be any element of $T_{\text{last}}(p)$ violating ii. Then $t$ is as shown on the left below. Introduce the null-transition $\lambda$ (i.e. $f_{\text{op}}(\lambda) = \lambda$) as shown on the right and re-arrange the input and output vertices of $t$ as shown.

$$
\begin{array}{c}
\cdots \quad t \quad \cdots \\
\cdots \quad \lambda \\
\cdots
\end{array}
\quad \Rightarrow 
\begin{array}{c}
\cdots \quad t \quad \lambda \\
\cdots
\end{array}
$$

For step 1. above, note that in the output net the place $y$ is blank, while the marked place $u$ is not in $P^{-1}[\text{SN}(p)]$. That the transformation is valid may be seen at once by considering any
marking on the original net which makes t firable, and noting that a corresponding marking exists on the transformed net.

2. Now let transformation T2 be as follows:

T2 input: i. linear FCP net G=(T,P,E)
   ii. place p ∈ P s.t. SN(p) satisfies the input and output conditions of T1

output: net G modified so that
   i. SN(p) contains a t-complete subgraph which is a basic FCP net containing p and p'
   and ii. the transformation is valid

step 1. If \(|T_{last}(p) \cap P^{-1}[SN(p)]| > 1\) then carry out the following:

Replace input edges and input places to \(T_{last}(p)\), shown on the left below, by the vertices and edges shown on the right.

Here \(k=|T_{last}(p) \cap P^{-1}[SN(p)]|\) and \(1=|T_{last}(p)|\).

To prove that the output net is a linear FCP net we will show that it satisfies the SM-reduction criterion. Note that for any SM-allocation AS on the original net there are k SM-allocations on the output net, because of the k possible values of an allocation at \(\lambda_1\) on the right above. For any SCSM-component S of the original net containing p, the output net will therefore have k different components. An application of Lemmas 4.2.2-4.2.4 yields the required result.
To show that the output net simulates the original, we must show that its set of firing sequences is a sufficiently large subset of the set of firing sequences of the original net, when instances of the null-transitions $\lambda_1$ and $\lambda_2$ are deleted. But the firing sequences of the output net are exactly those of the original which are obtained if we delay the firing of any $t \in T_{\text{last}}(p)$ till all the transitions of $T_{\text{last}}(p)$ are firable simultaneously. Clearly this is a sufficiently large subset to satisfy condition ii of Def.2.3.2 for any interpretation on $G$.

In view of the above arguments, it follows that $T2$ is a valid transformation.

$T2$ may be described as 'pinching together' the transitions in $T_{\text{last}}(p)$, thereby constraining them so that they can only be enabled all at once. As a result of $T2$, $SN(p)$ contains a basic FCP net as a subgraph with $p$ and $p'$ being the initial and terminal vertices, respectively, as diagrammed below. (Note that $p'$ is the place introduced into $SN(p)$ during $T2$.)

![Diagram of SN(p) and T_{last}(p)]

Note that the 'pinch' is necessary only if, in the original net, $\left| T_{\text{last}}(p) \cap P^{-1}[SN(p)] \right| > 1$. Also, note that $T2$ may involve the introduction of additional constraints into the output net, i.e. the set of firing sequences of the output net (after the deletion of null-transitions) may be smaller than that of the original net. In finding a top-down program simulating any given parallel program, therefore, an application of $T2$ may cause a
loss in the degree of parallelism attainable. This loss of parallelism (which may also be caused by transformations A2 and A3 of Appendices II and III respectively) is analyzed fully in Chapter 6, where we determine conditions under which it is unavoidable.

The following example illustrates the transformation T2:

**Ex. 4.2.6** Consider the net G and place p as shown below:

Here SN(p) is shown in bold lines, and \( T_{\text{last}}(p) = \{ t_1, t_2 \} \). On applying T2 we obtain the following net:

SN(p) is again in bold lines, and \( T_{\text{last}}(p) \) is unchanged. 

The intuitive need for T2 in structuring a linear FCP net may be seen from the above example.

We now describe the last of the three transformations of this Section.

2. For a place p in linear FCP net G, consider the case in which SN(p) does not wholly contain a Trace R of G, and FC(G) \( \cap P^{-1}[\text{SN}(p)] \neq \emptyset \). By Lemma 4.2.1 we therefore know that the blank paths within SN(p) define a partial order on the set of vertices of SN(p). We can therefore choose an FC place q in SN(p) s.t. SN(q) does not contain any FC places other than q, i.e. FC(G) \( \cap P^{-1}[\text{SN}(q)] = \{ q \} \). For an FC place such as q, we define the
transformation T3 as follows.

**T3 input:**

1. linear FCP net \( G = (T, P, E) \)
2. FC place \( q \) s.t. \( SN(q) \) does not contain a Trace
   R of \( G \), and \( FC(G) \cap P^{-1}[SN(q)] = \{q\} \); also, \( SN(q) \)
   satisfies the output conditions of T1 and T2

**output:** net \( G \) modified so that

1. \( \forall z \in P^{-1}[SN(q)] - \{q\} - T_{last}(q), \ z \in T^{-1}[SN(q)] \)

and ii. the transformation is valid

**step 1.** (a) Let \( z_1, \ldots, z_n \) be places in \( SN(q) \) s.t. \( Z = \{z_1, \ldots, z_n\} \)

\[ Z = \left\{ z \in P^{-1}[SN(q)] \mid z \notin T^{-1}[SN(q)] \cup \{q\} \cup T_{last}(q) \right\} \]

(b) Since there are no blank cycles in \( SN(q) \), assume

w.l.g. that \( \forall z_i, z_j \in Z, \)

\[ \left[ \begin{array}{c} \text{there exists a blank path in} \end{array} \right] \Rightarrow j < i \]

**step 2.** For \( i = 1, \ldots, n \) carry out the following:

2.1 find \( SN(z_i) \)

2.2 by an appropriate application of transformation A2

   of Appendix II, remove the pipe-line branches from

   \( SN(z_i) \)

2.3 construct a net \( \psi[SN(z_i)] \) isomorphic to \( SN(z_i) \)

2.4 delete the edges \( z_i \cap P^{-1}[SN(q)] \times \{z_i\} \)

2.5 add edges from the transitions \( z_i \cap P^{-1}[SN(q)] \)

   to \( \psi(z_i) \) in \( \psi[SN(z_i)] \)

2.6 identify the places \( \psi[T_{last}(z_i)] \) in \( \psi[SN(z_i)] \)

   with the places \( T_{last}(z_i) \) in \( SN(z_i) \)

**T3** in effect consists in making two copies of \( SN(z_i) \), for

each \( z_i \), one of them being connected to the input vertices of

\( z_i \) outside \( SN(q) \). The process may be thought of as replication
or 'splitting' of each of the subgraphs $SN(z_1)$ into two copies. This may be thought of as a generalization of the 'node-splitting' known for sequential program graphs. After presenting an illustrative example of T3, we shall give its proof of correctness.

**Ex. 4.2.7** Let $SN(q)$ be as shown below, where $t_2 \notin T^{-1}[SN(q)]$, $T_{last}(q) = \{ t_3 \}$ and $Z = \{ z_1 \}$. $SN(z_1)$ is as shown in bold lines.

Transformation T3 then yields the following subgraph.

To see that T3 is a valid transformation, consider any SCSM-component S of the original net. If S is disjoint with $SN(q)$ then T3 leaves S unmodified. If S is wholly contained in some $SN(z_1)$ then note that (since $SN(z_1)$ is a marked graph) S represents pipe-line branches within $SN(z_1)$. We eliminate these pipe-line branches from $SN(z_1)$ by applying transformation A2 to it, and therefore in the output net there is no SM-allocation corresponding to S. Finally, if S contains $z_1$ but is not wholly contained in $SN(z_1)$ then S must contain precisely one element of $T_{last}(z_1)$, by Lemma 4.2.2. But then T3 transforms S into an SCSM-component of the output net having the same number of tokens as S. Therefore, since all possible SM-allo-
cations of the output net have been accounted for, by SM-reduction criterion we have that the output net is a linear FCP net.

To see that the output net simulates the original, first note that for any \( i \in \{1, \ldots, n\} \), at the end of sub-step 2.2 we have a net simulating the original, since we know A2 to be a valid transformation. Now consider any marking on this net under which \( z_i \) is marked. Since \( SN(z_i) \) and \( \psi[SN(z_i)] \) are isomorphic, the transitions firable in this net are also firable in the net obtained at the end of sub-step 2.6. Also both these marked graphs terminate with the set of places \( T_{\text{last}}(z_i) \). With appropriate assignment of operators to the vertices of \( SN(z_i) \) and \( \psi[SN(z_i)] \), therefore, the resulting program will simulate the original. [Note that the loss of parallelism described in connection with T2 above may also be present when A2 is applied to \( SN(z_i) \).]

The above arguments show T3 to be a valid transformation.

Description of the transformations T1-T3 on the subgraphs \( SN(\cdot) \) satisfying the appropriate conditions is now complete. Examples 4.2.6 and 4.2.7 give some idea of their application later, when we present an algorithm which finds a top-down program simulating a given parallel program.

4.3 Trace Inputs and Outputs

In Section 4.1 we analyzed the structure of a Trace in a linear FCP net, and in Section 4.2 above we analyzed the structure of a linear FCP net in terms of the partial order \( \sigma_{SN} \)

**Transformation A2 is also used in the algorithms of Chapter 5 and is described in Appendix II.**
defined on the set of its Traces. In this Section we shall investigate the structure of the 'input' and 'output' vertices of a Trace, i.e. those places on the Trace through which tokens enter and leave the Trace. Recall that, in a linear FCP net under initial marking, only those Traces which contain 'start' are live — when the return link is added.

The following example illustrates these ideas, before we describe them formally.

Ex.4.3.1

Let $G$ be the linear FCP net shown above, and let $R$ be its Trace shown in bold lines. Then $\{p_1, p_2\}$ will be defined as the set of 'fundamental input places' of $R$; the place $p_4$ will not be in this set. Similarly, $\{p_3\}$ is the set of 'fundamental exits' of $R$, and does not include $p_6$. ##

We now present the formal definitions required.

Def.4.3.1 Let $R = (T_R, P_R, E_R)$ be a Trace in linear FCP net $G$. The set of fundamental input places of $R$, denoted $X^R_I$, is the set

$$X^R_I = \left\{ p \in P_R \mid \right\} \begin{array}{l}
\text{there is a path } \varPi \text{ in } G \text{ from 'start' to } p \text{ s.t.} \\
i. \text{ } p \text{ is the only vertex of } \varPi \text{ on } R, \text{ and} \\
ii. \text{ under the initial marking, 'start' is} \\
\text{the only marked place on } \varPi \end{array}$$

The following properties can then be shown for the set $X^R_I$. Following these (Lemmas 4.3.1 & 4.3.2) we shall make some further relevant definitions.
Lemma 4.3.1 Let \( R \) and \( X^R_I \) be as defined, and let \( p \in X^R_I \). Then \( p \) lies on a blank cycle in \( R \) under the initial marking \( M^0 \).**

Proof Consider a firing sequence \( \sigma \) of \( G \) which fires every transition on the path \( \tau \) from 'start' to \( p \) given by the defining condition of \( X^R_I \). Such a firing sequence leaves a token on \( p \).

Now by firing any transitions in \( \tau - p \) which become firable, in such a way that FC places on \( R \) fire 'into' \( R \), we reach a marking under which \( R \) is live and \( p \) is marked. Since \( G \) is a safe net, the token on \( p \) is the unique token on some cycle \( \tau \) in \( R \). Since \( p \) was initially blank, and since FC places on \( R \) were fired 'into' \( R \), it follows that \( \tau \) is blank under the initial marking and the result follows. **

Lemma 4.3.2 Let \( R \) and \( X^R_I \) be as defined. Any cycle \( \tau \) of \( R \) which is blank under the initial marking** contains a fundamental input place \( p \) of \( R \).

Proof Consider a 1-SCSM component of \( G \) containing \( \tau \). Let \( S \) be this component and let \( p_m \) be the marked place on \( S \) under \( M^0 \).

Then clearly a path \( \alpha_1(p_m, p) \) exists in \( S \) s.t. \( p \) is the only vertex of \( \alpha_1 \) on \( R \). Also, since \( t_0 \) is the only transition of \( G \) firable under the initial marking, a path \( \alpha_2('start', p_m) \) exists under the initial marking s.t. 'start' is the only marked place on \( \alpha_2 \). But then the composition of \( \alpha_2 \) with \( \alpha_1 \) satisfies Def. 4.3.1 and the result follows. **

The above two Lemmas give us some insight into the properties of \( X^R_I \) for a Trace \( R \) of \( G \). The next definition will help us identify the pipe-line branches (ref. Section 2.1) in a Trace.

**For a basic Trace of \( G \), \( X^R_I \) = \{'start'\} and we should replace 'blank cycle' above by 'elementary path from 'start' to 'end' containing exactly the one token on 'start'"
Def. 4.3.2 Let $R$ and $X^R_I$ be as defined above. The fundamental loop $\overline{R}_I$ of $R$ is defined as the union of all blank cycles in $R$ if $R$ is not basic, and as the union of all elementary paths in $R$ from 'start' to 'end' otherwise.

Ex. 4.3.2 In the net of the previous example, with $R$ as shown, $\overline{R}_I$ contains all transitions of $R$ except $t$, which is seen to lie on an initially marked cycle in $R$. In general, every transition in $T_R^-T^{-1}(\overline{R}_I)$ is on a pipe-line branch in $R=(T_R,P_R,E_R)$.

Def. 4.3.3 Let $R=(T_R, P_R, E_R)$ be a Trace in linear FCP net $G$. The nodes of $R$ are the places of $R$ in the set $X^R_n$ given by the following:

$$X^R_n = \left\{ p \in P_R \mid C(R,p) \text{ contains every blank cycle} \right\}$$

The outputs of $R$ are the places of $R$ in the set $X^R_o$ given by $X^R_o = X^R_n \cap FC(G)$.

For the graph of the previous examples, we have $X^R_n = \{ p_3, p_7, p_4, p_5 \}$ and $X^R_o = \{ p_3, p_5 \}$.

Def. 4.3.4 Let $R$ and $G$ be as above. The set of fundamental exits $X^R_e$ of $R$ is the set

$$X^R_e = \left\{ p \in P_R \mid \text{there is an elementary path } \Pi \text{ in } G \text{ from } p \text{ to 'end' s.t. } p \text{ is the only vertex of } \Pi \text{ on } R \right\}$$

For the graph of the previous examples we have $X^R_e = \{ p_3 \}$.

We now present a property of the Trace $R$ which links up the sets $X^R_n, X^R_o, X^R_e$ in a natural way.

Lemma 4.3.3 i. $X^R_e \neq \emptyset$

ii. $X^R_e \subseteq X^R_o \subseteq X^R_n$

Proof i. Follows from the fact that there is a path in $G$ from
every vertex to 'end'.

ii. That \( x_{o}^{R} \subseteq x_{n}^{R} \) follows at once from the definition. To show that \( x_{e}^{R} \subseteq x_{o}^{R} \), consider the opposite, i.e. let \( p \in x_{e}^{R} - x_{o}^{R} \). Then there exists a blank cycle \( \tau \) in \( R \) s.t. \( p \) is not on \( \tau \), and there is an elementary path \( \pi \) in \( G \) from \( p \) to 'end' satisfying the definition of \( x_{e}^{R} \). In view of Lemma 4.3.2 we can now find an elementary path \( \alpha \) in \( G \) from 'start' to some place \( q \) on \( \tau \) s.t. \( q \) is the only vertex of \( \alpha \) on \( R \). The following paths therefore exist in \( G \):

The bold lines in the above diagram indicate paths in \( R \).

But the above paths violate Lemma 1.4.4 when we consider that a strongly connected live and safe FCP net should result from \( G \) on the addition of a return link from 'end' to 'start'. The result follows in view of this contradiction.  

The net of the previous examples is seen to verify the above relations.

Def. 4.3.5 The set of secondary outputs of \( R \) is the set \( \text{FC}(G) \cap \left[ P_{R} - x_{o}^{R} \right] \).

This completes our analysis of the 'inputs' and 'outputs' of a Trace in a linear FCP net. Our analysis of the various structural features of linear FCP nets, as derived from the two reduction criteria of Hack (ref. Chapter 1) is also complete. In the next Chapter we shall use these properties to develop an algorithm which finds a top-down program simulating a given program.
CHAPTER 5

STRUCTURE ALGORITHM

Based on the analysis in the previous Chapter of the structure of linear FCP nets, and the earlier definitions of the behaviour of parallel programs, etc., we present in this Chapter an algorithm which finds a top-down program simulating any given parallel program. Thus we shall prove the (intuitively obvious) result that there exists a top-down program simulating any given parallel program. Also, the algorithm of this Chapter will be used in proving a part of the results of Chapter 6.

The algorithm we present is considerably more complex than previously used transformations which establish the relative powers of representation of various classes of 'Structured' sequential programs.\(^{(15,18)}\) One major difference is brought to light, moreover, when we consider parallel programs in the context of top-down design: it is found that there are parallel programs for which there exist no top-down equivalents. In other words, any parallel program simulating one such parallel program must involve additional constraints between operators — constraints which were not present in the original program. In Chapter 6 we shall present conditions under which there is no such loss of parallelism in finding a top-down program which simulates a given parallel program.

Section 5.1 below outlines two basic, complementary techniques called reduction (or representation) and replacement which will be used in the algorithm, which is in Sections 5.2 - 5.4. The proof of correctness of each part of the algorithm is

\(^{**}\) See foot-note on p.71 below.
presented immediately following it. Section 5.5 presents an example of the transformation carried out by the algorithm.

5.1 Outline

In this Section we introduce two techniques: reduction (or representation) and replacement, which will enable us later to describe the Structure Algorithm as a recursive procedure. This will simplify the description of the algorithm as well as a proof of its correctness. We shall prove in this Section that, given the necessary conditions for their application, the above two techniques give the desired results.

The algorithm of this Chapter has the following general form:

(a) we identify a reducible subgraph $G^*$ of the given net $G$ and reduce it, representing it by a single transition $t^*$ or a transition-place pair $(t^*, x^*)$,

(b) we apply the algorithm recursively to the reduced net, and (c) in the net resulting from (b) above, we replace every occurrence of $t^*$, or the pair $(t^*, x^*)$, by $G^*$.

The precise definitions of 'reducible', 'reduce', 'replace' etc. will be made below. In Sections 5.2 - 5.4, then, we shall refer to these techniques and employ them in the algorithm.

A. Reduction/Representation

This step takes one of two alternate forms, denoted here type#1 and type#2, depending on which of the two permissible forms the subgraph $G^*$ in question has.

**type#1** In the linear FCP net $G$, let the subgraph $G^* \neq G$ have the following form:
and further, let \( G^* \) satisfy the following condition:

\[
\text{condition 5.1.1} \quad \begin{cases} 
\text{i. no Trace } R \text{ of } G \text{ is a subgraph of } G^*, \\
\text{ii. every transition } t \text{ in } G^* \text{ is on an elementary path from } x_1 \text{ to } x_2
\end{cases}
\]

Note that i. states, in effect, that \( G^* \) is a basic FCP net.
And ii. has the significance that it rules out any transitions being on the pipe-line branches in \( G^* \). As the following example illustrates, in a linear marked graph any transition which is not on an elementary path from the initial to the terminal vertex is on a pipe-line branch. (See also Section 2.1.)

Ex. 5.1.1 Consider the following marked graph:

\[
\begin{array}{c}
\circ \rightarrow \circ \rightarrow \circ \rightarrow \circ \rightarrow \circ \rightarrow \circ \\
\circ \rightarrow \circ \rightarrow \circ \rightarrow \circ \rightarrow \circ \rightarrow \circ
\end{array}
\]

Transition \( t \) above is so situated that it may fire after the firing of \( t' \).

We shall see that in general we must rule out pipe-line branches from \( G^* \) in order to be able to represent it by a single transition. However, two exceptions to this rule will be mentioned later.

Now reduction of \( G^* \) to a single transition \( t^* \) consists in \( G^* \) being substituted by the following

\[
\begin{array}{c}
x_1 \rightarrow t^* \rightarrow x_2
\end{array}
\]

Alternatively, we say that the subgraph \( G^* \) is here represented by the transition \( t^* \).
To obtain the reduced FPP, we associate with $t^*$ a new operator $f^*$ which, intuitively, will represent the computation carried out by $G^*$. Clearly, for a non-trivial subgraph $G^*$, we have obtained a net (or an FPP) which is smaller than the original.

**type#2** In the linear FCP net $G$, let a strongly connected subgraph $G^*$ exist which is of the following form:

![Diagram](image)

and further, with $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$, let $G^*$ satisfy the following condition:

\[
\text{condition 5.1.2} \quad \begin{cases} 
\text{i. for each Trace } R = (T_R, P_R, E_R) \text{ of } G \text{ in } G^* \\
T_R = T^{-1}(\overline{R}_I) \\
\text{ii. for every SCSM-component } S \text{ of } G, \text{ either } (X \cup Y) \cap P^{-1}(S) = \emptyset \text{ or } (X \cup Y) \subseteq P^{-1}(S)
\end{cases}
\]

i. above corresponds to ii. of condition 5.1.1, since any $t \in T_R$ which is not on $\overline{R}_I$ is on a pipe-line branch of $R$. (See Section 4.3) While ii. in the above condition states that in any Trace $R$ in $G^*$, all elements of $X \cup Y$ should be at the same 'level of parallelism'. (See Section 4.1) Note that $G^*$ above has $m$ places $y_1, \ldots, y_m$ which have one or more input transitions outside $G^*$, and $n$ places $x_1, \ldots, x_n$ which have one or more output transitions outside $G^*$.

We then **reduce** $G^*$ of type#2 to a transition-place pair $(t^*, x^*)$ as shown below, where $k = \sum_{i=1}^{n} |x_i - T^{-1}(G^*)| :$
As before, we alternatively say that the pair \((t^*, x^*)\) represents \(G^*\). To obtain the reduced FPP, we associate with \(t^*\) a new operator \(f^*\), and with \(x^*\) a new predicate \(g^*\). As in the previous type of reduction, the resulting net is smaller than the original.

B. Replacement

Let \(G\) be a linear FCP net with a subgraph \(G^*\) of either type, and let \(G'\) be obtained from \(G\) by means of a single step of reduction. Let the corresponding FPPs be \(F\), \(F'\) and let \(f^*\), \(g^*\) etc. be as defined above.

Now let \(F''\) be a top-down FPP simulating \(F'\) which satisfies the following:

\[
\begin{cases}
\text{if } G^* \text{ is a type#2 subgraph then every occurrence of } f^* \text{ and } g^* \text{ in } F'' \text{ is in the configuration shown below} \\
\end{cases}
\]

\[
\begin{array}{c}
\text{1} \\
\vdots \\
\text{k}
\end{array}
\]

\[
\begin{array}{c}
f^* \\
g^*
\end{array}
\]

Note: It can trivially be shown, with \(F', F''\) etc. as above, that \(|f^{-1}_{op}(f^*)| > 1\); and \(|f^{-1}_{pr}(g^*)| > 1\) for a type#2 reduction.

Let \(F^5\) be the FPP obtained from \(F''\) by replacing every occurrence of \(f^*\) or \((f^*, g^*)\), depending on the type of reduction, in \(F''\) with \(G^*\), with the appropriate input and output vertices and edges.

The following diagrams illustrate the relationships between \(F\), \(F'\), \(F''\) and \(F^5\) for the two types of reductions:
In the next few theorems we show that the techniques of reduction and replacement work correctly — by showing that $\tilde{f}^s$ has the desired properties — and that therefore their application in the structuring algorithm is justified.

**Thm. 5.1.1** Let $G$ be a linear FCP net and let $G'$ be obtained from $G$ by a single reduction step. Then $G'$ is a linear FCP net.

**Proof** We shall show that $G'$ satisfies the SM-reduction criterion; since 'start','end' etc. are unchanged in $G'$, the result will follow. Therefore consider an SCSM-component $S$ of $G$. If $S$ is disjoint with $G^*$ then it is a valid component in $G'$. If $S$ is wholly contained in $G^*$ then $G'$ has no corresponding SM-allocation or SCSM-component. Finally therefore, consider the case in which $S$ is not disjoint with $G^*$, nor is wholly contained in $G^*$.

Then $S$ has the form of the graph on the left below, where
k=m=1 for a type#1 reduction. Also note that part ii. of condition 5.1.2 is required for type#2 reductions. But then the reduction step causes $S_1$, the subgraph of $S$ in $G^*$, to be reduced as on the right. Also, from the analysis of the previous Chapter we know that $S_1$ is blank. Therefore the number of tokens on the SCSM-component is unchanged. The modified component is therefore a valid component in $G'$.

In view of the above three possibilities for an arbitrary component of $G$, we see that $G'$ satisfies the SM-reduction criterion and the result follows. 

\textbf{Thm.5.1.2} Let $G^S$ and $G''$ be as defined above, i.e. $G^S$ is obtained from $G''$ by replacement. Then $G^S$ is a linear FCP net.

\textbf{Proof} Follows from arguments similar to those in Thm.5.1.1 above.

To simplify the next proof, we now introduce two onto mappings, $\rho : \text{FS}(G) \rightarrow \text{FS}(G')$ and $\rho^S : \text{FS}(G^S) \rightarrow \text{FS}(G'')$, for the case of a type#1 reduction.

(a) Let $\alpha \in \text{FS}(G)$. Then $\rho(\alpha) \in \text{FS}(G')$ is found by replacing any occurrences of elements of 'x_2 in $\alpha$ by $t^*$, and deleting any occurrences of the other transitions of $G^*$ in $\alpha$. (Recall that $x_2$ is the terminal vertex of $G^*$.) Intuitively, we record one occurrence of $t^*$ in $\rho(\alpha)$ for every complete firing of the net $G^*$.

(b) To define $\rho^S$ let us denote by $t^*, i$, $0 \leq i < \left| f^{-1}_{op}(f^*) \right| -1$, the elements of $f^{-1}_{op}(f^*)$ in $G''$. Let us then denote by $G^*, i$, $0 \leq i < \left| f^{-1}_{op}(f^*) \right| -1$, the corresponding occurrences of $G^*$ in $G^S$. Finally let us denote by $x_1^i$ and $x_2^i$, $0 \leq i \leq \left| f^{-1}_{op}(f^*) \right| -1$, the
initial and terminal vertices of $G^*, i$, respectively.

Now let $\alpha \in FS(G^S)$. Then $\rho^S(\alpha) \in FS(G^r)$ is found by replacing any occurrences of elements of $x^i_2$ in $\alpha$ by $t^*, i$, and deleting any occurrences of the other transitions of $G^*, i$ in $\alpha$, for $0 \leq i \leq |f^{op}_{op}^{-1}(f^*)| - 1$. As in the case of $\rho$, we record one $t^*, i$ for every complete firing of the net $G^*, i$.

**Thm. 5.1.3** Let $F$, $F'$, $F''$, and $F^S$ be as defined above for a type#1 reduction. Then:

i. $F^S$ simulates $F$, and

ii. if $F''$ is equivalent to $F'$, then $F^S$ is equivalent to $F$.

**Proof** (i) We first show that $\beta(F^S) \subseteq \beta(F)$, i.e. for an arbitrary behaviour sequence (b.s.) of $F^S$, we show that it belongs to $\beta(F)$. Therefore let $\alpha$ be an arbitrary firing sequence (f.s.) of $G^S$ and let $\beta(\alpha)$ be the corresponding b.s. of $F^S$.

From our construction of $\rho^S$ it follows that: if any transition in $G^*, i$ is firable under the marking $\delta(M^0, \alpha)$, then $t^*, i$ is firable in $G^r$ under the marking $\delta(M^0, \rho^S(\alpha))$, for $0 \leq i \leq |f^{op}_{op}^{-1}(f^*)| - 1$. Also, under that condition, no transition in any other $G^*, j$, $j \neq i$, is firable, in view of part i. of condition 5.1.1. Finally, any other transitions of $G^S$ firable under $\delta(M^0, \alpha)$ are firable in $G^r$ under $\delta(M^0, \rho^S(\alpha))$.

Then $\beta(\rho^S(\alpha))$ is the b.s. of $F''$ corresponding to $\rho^S(\alpha)$, and $\beta(\rho^S(\alpha)) \subseteq \beta(F')$ since $F''$ simulates $F'$. Therefore $\exists \sigma' \in FS(G')$ s.t. $\beta(\sigma') = \beta(\rho^S(\alpha))$ and the set of operators in $G^r$ enabled under the marking $\delta(M^0, \rho^S(\alpha))$ are enabled in $G'$ under the marking $\delta(M^0, \sigma')$. And, using a similar argument, $\exists \sigma \in FS(G)$ s.t. $\rho(\sigma) = \sigma'$.

In general there will be more than one possible choices
for \( \sigma \) above, since different partial firings of \( G^* \) can cause \( \sigma \) to map into the same sequence \( \sigma' \) of \( G' \). We choose that firing sequence \( \sigma \) for which the partial firing of \( G' \) in \( G \) corresponds to the firing of the appropriate \( G^* \), \( i \) in \( G^S \). That such a firing sequence \( \sigma \) exists can be shown easily by induction on the number of occurrences of transitions of \( G^* \), \( i \) in \( \alpha \). Using the same argument, it follows in a straightforward manner that the above sequence \( \sigma \) satisfies \( \beta(\sigma) = \beta(\alpha) \), from which it follows that \( \beta(F^5) \subseteq \beta(F) \).

Note that the above argument holds only if part i. of condition 5.1.1 is satisfied, since it ensures that on the firing of a transition in \('x_2\) no transition in \( G^* \) is firable. This means that the marking over \( G^* \) is then uniquely determined.

Now we must show that no equivalence class of \( \text{sim} \) on \( \beta(F) \) is disjoint with \( \beta(F^5) \). Let \( \sigma \in FS(G) \) be an arbitrary f.s. of \( G \). Then we know that \( \exists \sigma' \in FS(G') \) s.t. \( (\beta(\sigma'), \beta(F(\sigma))) \in \text{sim} \) and \( \beta(\sigma') \subseteq \beta(F''), \) since \( F'' \) simulates \( F' \). Let \( \alpha'' \in FS(G'') \) s.t. \( \beta(\alpha'') = \beta(\sigma') \). As before, but with roles of \( F^5 \) and \( F \) reversed, we know that \( \exists \alpha \in FS(G^S) \) s.t. \( \beta(\alpha) = \alpha'' \). The rest of the argument follows as before, with roles of \( F^5 \) and \( F \) reversed. Induction on the number of occurrences of transitions of \( G^* \) in \( \sigma \) yields \( (\beta(\sigma), \beta(\alpha)) \in \text{sim} \), i.e. no equivalence class of \( \text{sim} \) on \( \beta(F) \) is disjoint with \( \beta(F^5) \).

Both clauses of Def. 2.3.2 are therefore satisfied, i.e. \( F^5 \) simulates \( F \).

(ii) To show that \( \beta(F^5) = \beta(F) \) in this case, we proceed exactly as in the latter part of (i) above, but make use of the stronger premise that \( \beta(F') = \beta(F'') \). Each of the arguments
applies as before to yield $\beta(\mathcal{F}) \subseteq \beta(\mathcal{F}')$. Combined with the first part of (i), we then get $\beta(\mathcal{F}) = \beta(\mathcal{F}')$, i.e. $\mathcal{F}$ and $\mathcal{F}'$ are equivalent.

Note that the above proof depends on the fact that we ruled out pipe-line branches from $G^*$ (part ii of condition 5.1.1). However, note that we dispense with part ii of condition 5.1.1 if any of the following two are satisfied:

(a) $|f^{-1}_{op}(f^*)| = 1,$

(b) input and output transitions of $G^*$ appear only on basic Traces of G (see Section 2.1).

This is so because in each of the two cases (a) and (b) above, transitions in at most one $G^*,i$ in $G^S$ can be enabled at one time.

Now consider type#2 reductions. Once again we define two onto mappings, $f:FS(G) \rightarrow FS(G')$ and $\rho^*:FS(G^S) \rightarrow FS(G'')$. Let $X = \{x_1, \ldots, x_n\}$ as before.

(a) Let $\alpha \in FS(G)$. Then $f(\alpha) \in FS(G')$ is found by: i. replacing with $t^*$ any occurrences of elements of $\cdot X$ which satisfy the condition that the next occurrence in $\alpha$ of a transition in $X$ is not in $G^*$, and ii. deleting any other occurrences of transitions of $G^*$ in $\alpha$. As before, we have recorded one $t^*$ in $\rho(\alpha)$ for every complete firing of $G^*$.

(b) We define $\rho^*:FS(G^S) \rightarrow FS(G'')$ in a similar manner, letting $G^*,i, 0 \leq i \leq |f^{-1}_{op}(f^*)| - 1$, denote all the occurrences of $G^*$ in $G^S$.

Each of the arguments of Thm. 5.1.3 applies then, with little or no change, giving us the following:
Thm. 5.1.4 let $F$, $F'$, $F''$, and $F'''$ be as defined for a type#2 reduction. Then:

i. $F'''$ simulates $F$, and

ii. if $F''$ is equivalent to $F'$, then $F'''$ is equivalent to $F$. #

The conditions described above, under which part ii of condition 5.1.1 may be relaxed, now apply to part i of condition 5.1.2 for a type#2 reduction.

Apart from the above-mentioned properties of reduction and replacement, we now show another — one that refers to the possibility of transforming $G'$ and $G^*$ separately into top-down graphs, and then bringing the two together. This property will provide us with the basis for a recursively designed algorithm to find a top-down program simulating a given program $F$.

Def. 5.1.1 Let $G^*$ be as defined above for either type of reduction. We say $G^*$ is a top-down subgraph of $G$ if it can be described as a state machine possibly modified by one or more S- or M- substitution steps. #

Ex. 5.1.1 Consider the following two subgraphs of some net $G$.
Here (a) is type#1, while (b) is type#2:

(a)

(b)

Then (a) is not a top-down subgraph, while (b) is one. This may be verified at once from the definitions of S- and M-substitutions in Chapter 3. #
In view of the above definition, we now state the relevant theorem which establishes a useful property of reduction and replacement.

**Thm. 5.1.5** Let $F', F'', F^s$ as defined for either type of reduction. If $G^*$ is a top-down subgraph of $G$ then at most a trivial modification of $G^S$ ensures that it is a top-down net.

**Proof** We know that $F''$ is the top-down FPP which simulates the reduced FPP $F'$, and satisfies condition 5.1.3. The result follows from the fact that each of the substitution steps needed to obtain $G^*$ is a valid substitution in $G^S$. A small modification may be required when a $t^*, i$ is introduced into $G''$ through an M-substitution, and $G^*, i$ is then re-drawn as shown below with two $\lambda$-transitions:

This ensures that each of the substitutions of $G^*, i$ is a valid substitution in $G^S$. The result then follows from the definition of top-down nets.

We are now ready to describe the structuring algorithm, in the next three Sections, in which we shall employ the techniques of reduction and replacement.

5.2 Structure Algorithm - I

As mentioned in the above Section, the algorithm we now describe is recursive. We start processing the given net $G$ at a subgraph which is in some sense the smallest subgraph which can suitably be transformed. Alg. 5.2.1 finds this subgraph in
the net G. Definitions 5.2.1-5.2.3 lead up to this algorithm. Alg. 5.2.2 is then the algorithm which is first invoked to find a top-down program simulating a given program.

**Def. 5.2.1** A cluster $G^C$ in linear FCP net $G$ is a strongly connected subgraph of $G$ which can be described as the union of some Traces $R_1, ..., R_n$, $n \geq 1$, of $G$.

**Ex. 5.2.1** Consider the following net:

In bold lines we have shown two disjoint clusters of $G$. $R_1, R_2$ and $R_3$ are the three non-basic Traces of $G$.

**Def. 5.2.2** A Trace $R$ of linear FCP net $G$ is said to be **innermost** if it is a minimal element of the poset $(R(G), \sigma_{SN})$.

In the graph of Ex. 5.2.1, each of the three non-basic Traces is an innermost Trace. Note that a basic Trace of $G$ is always a maximal element of the poset $(R(G), \sigma_{SN})$ and therefore is innermost if it has no Trace 'within' any of its parallel branches.

**Def. 5.2.3** Let $p$ be a place in linear FCP net $G$ s.t. $SN(p)$ contains one or more Traces of $G$. Then a **last cluster** in $SN(p)$ is a cluster $G^C$ of $G$ in $SN(p)$ satisfying the following:

i. there is no blank, elementary path $\pi(x, y)$ in $SN(p)$ from a place $x$ on $G^C$ to a place $y$ on another cluster $G^{C'}$ of $G$ in $SN(p)$ which is disjoint with $G^C$, and
ii. $G^c$ is not a proper subgraph of another cluster $G^{c''}$ of $G$ in $SN(p)$.

**Ex. 5.2.2** In the graph of the previous example, with $R_1, R_2, R_3$ as shown, $R_3$ forms a last cluster in $G = SN('start')$. However, $R_1$ is not a last cluster because it is contained in the cluster of the union of $R_1$ and $R_2$.

The following Lemma is needed as the basis for Alg. 5.2.1.

**Lemma 5.2.1** Let $G$ and $p$ be as in Def. 5.2.3 above. Then $SN(p)$ contains a last cluster.

**Proof** Since $SN(p)$ contains at least one Trace of $G$, there is at least one cluster $G^c$ of $G$ in $SN(p)$. If $G^c$ violates condition i of Def. 5.2.3, then we may choose $G^{c'}$ in place of $G^c$. Part ii of Def. 5.2.3 and Lemma 4.2.1 would then ensure that there is no blank, elementary path in $SN(p)$ from $G^{c'}$ to $G^c$. If, on the other hand, condition ii of Def. 5.2.3 is violated, then we can choose $G^{c''}$ in place of $G^c$. Thus in any event we can ensure that both parts of Def. 5.2.3 are satisfied and the result follows.

Using the above definitions, etc., we now present an algorithm which finds a cluster of innermost Traces in $G$. This cluster of innermost Traces is the subgraph at which our algorithm begins processing $G$. In view of Def. 5.2.3, note that this subgraph is defined only if $G$ is not a basic net. Basic nets will be transformed through a separate algorithm, as will become clear in the following pages.

**Alg. 5.2.1** input : non-basic linear FCP net $G$

output : a cluster of innermost Traces in $G$

step 0. Find a last cluster in $G = SN('start')$. Let this cluster
be 'current cluster'.

step 1. If all Traces in the 'current cluster' are minimal elements of \((R(G), \sigma_{SN})\) then go to step 4 below.

Otherwise let \(R\) be any Trace in the 'current cluster' which is not a minimal element of \((R(G), \sigma_{SN})\).

step 2. Let \(p_1, \ldots, p_k\) be all the secondary outputs of \(R\) (see Section 4.3), and w.l.g. assume that they are numbered so that \(C(R, p_i) \subset C(R, p_j) \Rightarrow i < j\).

step 3. For \(i = 1, \ldots, k\) carry out the following:

- If \(\left[ SN(p_i) \right] \) contains a Trace \(R'\) of \(G\) then
  - 3.1 Choose as 'current cluster' a last cluster in \(SN(p_i)\)
  - 3.2 Go to step 1 above.

step 4. The 'current cluster' is the desired cluster of innermost Traces. Denote it by \(G^c\).

The subgraph \(G^c\) of a linear net \(G\), as found by the above algorithm, will be transformed in Section 5.4 into a type#2 reducible subgraph of \(G\). An application of type#2 reduction to \(G^c\) (as defined in the previous Section), followed by a recursive call to the structuring algorithm (i.e. Alg.5.2.2 below), will constitute our essential strategy. That the algorithm satisfies condition 5.1.3 can be shown easily, for example, by induction on the number of applications of type#2 reduction required.

The following algorithm, Alg.5.2.2, is the one which is invoked first to find a top-down FPP simulating a given FPP.

All the other algorithms of this Chapter, including Alg.5.2.1

**'Algorithm' here sometimes refers to the whole strategy of Sections 5.2-5.4 comprising several 'procedures', and sometimes to an individual 'procedure'. The context usually clarifies the meaning.
above, are invoked subsequently under appropriate conditions. Recursive calls to Alg.5.2.2 (which is, strictly speaking the main 'Structure Algorithm') are also generated at appropriate points.

Recall the definition of a 'last cluster' in the previous Section. It follows at once from that definition that a basic FCP net has no 'last cluster' in it. In this Chapter, Section 5.3 presents an algorithm to structure a basic net. This algorithm, Alg.5.3.1, is invoked whenever a basic net is to be transformed. (See, for example, step 0 below.)

We are now ready to present Alg.5.2.2, which is invoked first to structure an arbitary program \( \mathcal{F} \).

**Alg.5.2.2**

Input : FPP \( \mathcal{F} = (G, S_{op}, f_{op}, S_{pr}, f_{pr}) \)

Output : top-down FPP \( \mathcal{F}' \) simulating \( \mathcal{F} \)

step 0. If \( G \) is a basic net then

0.1 Apply Alg.5.3.1 to \( G \)

0.2 Go to step 3 below.

step 1. Apply Alg.5.2.1 to find a cluster of innermost Traces \( G^c \) in \( G \).

step 2. Apply Alg.5.4.1 to \( G \) and \( G^c \).

step 3. The output of Alg.5.3.1 or Alg.5.4.1 (depending on whether this step is reached from step 0 or step 2, respectively) is the required program.

The overall strategy of Sections 5.2 - 5.4 may be seen in the above description of Alg.5.2.2, which invokes the other 'procedures', in this Section and the following two Sections. Recursive calls to Alg.5.2.2 are generated from Alg.5.3.1 and Alg.5.4.1 as appropriate. It will be seen that Alg.5.3.1 makes
use of only one of the two reductions of Section 5.1. This is so because Alg. 5.3.1 processes only basic nets, which cannot have a subgraph of type #2. Alg. 5.4.1 makes use of both types of reduction.

5.3 Structure Algorithm - II

The algorithm of this Section, Alg. 5.3.1, is invoked from Alg. 5.2.2 of the previous Section exactly in the case that the net $G$ is a basic net. In other words, input to Alg. 5.3.1 is a basic FCP net; the output of course is a top-down net simulating the input net.

We denote by $\overline{G} = (T, F, E)$ the net which is processed by Alg. 5.3.1, i.e. we let $\overline{G}$ serve as a 'formal parameter' for the algorithm. Before describing the algorithm, however, we present an example of the transformation carried out by it.

**Ex. 5.3.1** Consider the following net, clearly a basic net which is not top-down:

Then Alg. 5.3.1 yields the following top-down program simulating it, where $t_8^1, t_8^2, \ldots, t_{12}^1, t_{12}^2$ etc. are the respective operators $f_8, \ldots, f_{12}$.
We know that there are no blank cycles in a basic FCP net. (This may be seen, for example, by applying Lemma 4.2.1 to it.) The blank paths in \( \overline{\mathcal{G}} \) therefore define a partial order on the set of vertices in \( \overline{\mathcal{G}} \). We can therefore find an FC place \( p \) in \( \overline{\mathcal{G}} \) (if \( \text{FC}(\overline{\mathcal{G}}) \neq \emptyset \)) which satisfies the following:

condition 5.3.1 \[
\{ \forall q \in \text{FC}(\overline{\mathcal{G}}), q \neq p, \\
\sim \left[ \text{there is a blank path in } \overline{\mathcal{G}} \text{ from } p \text{ to } q \right] \}
\]

Alg. 5.3.1 below begins processing \( \overline{\mathcal{G}} \) at a place such as \( p \) above. In effect the algorithm proceeds backwards along the partial order of blank paths, till all the FC places in \( \overline{\mathcal{G}} \) have been accounted for.

Therefore for the remaining part of this Section let \( p \) be an FC place of \( \overline{\mathcal{G}} \) satisfying condition 5.3.1, and let \( p' = \{ t_1^p, \ldots, t_k^p \} \). Also let \( \text{SN}(p) \) be as defined in Section 4.2. Alg. 5.3.1 will be described in terms of the next two definitions.

Def. 5.3.1 Let \( \overline{\mathcal{G}}, p, \text{SN}(p) \) etc. be as above. The \( j \)-th alternate subgraph of \( \text{SN}(p) \), denoted \( \text{SN}(p)/j \), for \( 0 < j < |p'| \), is obtained in \( \text{SN}(p) \) by means of the following two-step procedure:

step 0. Delete the edges \( (p, t_k^p) \), for all \( k \in \{1, \ldots, |p'|\} \setminus \{j\} \), from \( \text{SN}(p) \).

step 1. Of the resulting subgraph of \( \text{SN}(p) \), find the largest component reachable from \( p \). It is the desired subgraph \( \text{SN}(p)/j \).

The above definition corresponds to choosing that subgraph of \( \text{SN}(p) \) which will be reached if the transition \( t_j^p \) fires when the FC place \( p \) is marked. The following example illustrates the above definition on the graph of Ex. 5.3.1.
Ex. 5.3.2 For the graph of the previous example, the subgraph \(SN(p)/t_4\) (using the above notation somewhat loosely) is shown below:

![Diagram](image)

The next definition is in a sense complementary to the first, in that we delete the edge \((p, t^p_j)\) and then find the largest component of \(\overline{G}\) (note: not \(SN(p)\)) reachable from the initial vertex of \(\overline{G}\).

**Def. 5.3.2** Let \(\overline{G}, p, SN(p)\) etc. be as above. We say \(SN(p)\) is deleted from \(\overline{G}\) by the following two-step procedure:

1. **step 0.** Delete the edge \((p, t^p_j)\) from \(\overline{G}\).
2. **step 1.** Of the resulting subgraph of \(\overline{G}\), find the largest component reachable from the initial vertex of \(\overline{G}\).

Ex. 5.3.3 For the graph of the previous examples, deleting \(SN(p)/t_4\) yields the following subgraph:

![Diagram](image)

Using the above definitions, we now describe the algorithm which yields a top-down program simulating the given basic program. Note the call to Alg. 5.2.2 at step 5 and the use of type#1 reduction at step 4. Note also that at the end of step 3 the subgraph \(SN(p)\) is a top-down subgraph of \(\overline{G}\).
Alg. 5.3.1 input: \( FPP \vec{F} = (\vec{G}, \vec{S}_o, \vec{F}_o, \vec{S}_r, \vec{F}_r) \) where \( \vec{G} \) is a basic net

output: top-down program simulating \( \vec{F} \)

step 0. If \( \vec{G} \) has no FC places, i.e. \( FC(\vec{G}) = \emptyset \), then by definition it is a top-down net. Go to step 7.

step 1. Otherwise, i.e. \( FC(\vec{G}) \neq \emptyset \), find an FC place \( p \) in \( \vec{G} \) satisfying condition 5.3.1.

step 2. Apply transformations \( T_1, T_2, T_3 \) of Section 4.2 to \( SN(p) \).

step 3. For \( k = 1, \ldots, |p'| \) carry out the following:

3.1 Delete \( SN(p)/k \) from \( \vec{G} \)

3.2 Add to the resulting net a disjoint net isomorphic to \( SN(p)/k \), with the appropriate operators, between \( p \) and the place \( p' \) introduced during \( T_2 \)

3.3 Remove the pipe-line branches from \( SN(p)/k \), which is a marked graph, by an application of transformation A2 of Appendix II.

step 4. Reduce \( SN(p) \) by means of a type#1 reduction to a single operator \( t^* \).

step 5. Apply the structuring algorithm (Alg. 5.2.2) to the reduced FPP.

step 6. Replace every occurrence of \( t^* \) in the resulting program by \( SN(p) \).

step 7. The resulting program is the desired top-down FPP. ##

That \( T_1, T_2, T_3 \) of Section 4.2 work correctly we have seen in that Section. We must show here that the net obtained at the end of step 3 is a linear net, and that the corresponding FPP simulates \( \vec{F} \).
To show that the net obtained at the end of step 3 is a linear net, we shall use the SM-reduction criterion as before. Consider any SCSM-component of the original net containing p, say S. Then S must contain an elementary blank path from each \( t_j^p \) to \( p' \). When \( SN(p)/j \) is deleted from \( \overline{G} \) (sub-step 3.1) and an isomorphic net added (sub-step 3.2), the effect in S is to delete a path \( \Pi(t_j^p, p') \) from S and add an isomorphic disjoint path from \( t_j^p \) to \( p' \). This modification in S is as diagrammed below:

Clearly therefore we obtain a valid component of the output net when S is modified as above. Since any SCSM-component of the original net which does not contain p is left unmodified, it follows that the SM-reduction criterion is satisfied and the output net is a linear basic net.

That the net at the end of step 3 simulates the original follows from the fact that the firing sequences of \( SN(p)/j \), \( j = 1, \ldots, |p'| \), as well as those of the net obtained on deleting the respective \( SN(p)/j \), are unchanged.

Finally, at the end of step 3 we have \( SN(p) \) which is a top-down subgraph of \( \overline{G} \) of type#1. By the arguments of Section 5.1, therefore, steps 4 - 6 are valid, and thus the output net of Alg.5.3.1 is the desired net, and the corresponding FPP simulates the original. The steps can be followed in Ex.5.3.1. Note that \( t_{12} \) has been absorbed just after \( t_{11} \) (through transformation A2 of Appendix II).
5.4 Structure Algorithm - III

Recall that in Alg. 5.2.2, if G is not a basic net, then we identify a cluster of innermost Traces $G^C$ in G and apply to it Alg. 5.4.1 of this Section (see step 2 in Alg. 5.2.2). Alg. 5.4.1 transforms $G^C$ into a type#2 reducible top-down subgraph of G, reduces it to a pair $(f^*, g^*)$, and then initiates a recursive application of the structuring algorithm on the reduced net. In the net resulting from this recursive application, each occurrence of $(f^*, g^*)$ is finally replaced by $G^C$.

Therefore let $G^C$ be as defined in Section 5.2, and let $R(G^C)$ denote the set of Traces of $G^C$.

Def. 5.4.1 Exits of $G^C$ are the FC places on $G^C$ which have one or more of their output transitions outside $G^C$; $X_E^C = \{x_1, \ldots, x_n\}$ denotes the set of exits of $G^C$. Inputs of $G^C$ are the places on $G^C$ which have one or more input transitions outside $G^C$; $X_I^C = \{y_1, \ldots, y_m\}$ denotes the set of inputs of $G^C$.

Ex. 5.4.1 In the net of Ex. 5.2.1, $y_1$ is the only input of the innermost cluster formed by $R_3$. $x_1$ is its only output.

Note that these definitions are parallel to those of the outputs and inputs of a Trace in G (see Section 4.3).

Def. 5.4.2 Let G and $G^C$ be as above. The remainder graph $G^r$ is then defined as the union of Traces in $R(G) - R(G^C)$ which satisfies the following conditions:

Let $R(G^r)$ denote the set of Traces of $G^r$, then

i. $R(G^r) \cap R(G^C) = \emptyset$, i.e. no Trace of $G^C$ appears in $G^r$,

ii. every element of $X_E^C \cup X_I^C$ is on $G^r$, and

iii. no Trace of G can be added to $G^r$ without violating i above.
can be constructed by considering the Traces of \( G \) one by one, and combining those which may be combined without violating \( i \) above, till both \( ii \) and \( iii \) above are satisfied. Intuitively, \( G^r \) is obtained from \( G \) by deleting from it the iterations represented by \( G^c \).

**Ex. 5.4.2** Consider the net \( G \) below, where \( G^c \) is in bold lines.

\( G^c \) contains three Traces, all the Traces not containing the initial vertex. And \( X^c_E = \{x_1, x_2\} \), \( X^c_I = \{y_1, y_2, y_3\} \). Then \( G^r \) will be as shown below.

As can be seen from the example, \( G^r \) is obtained from \( G \) by excluding from it the Traces of \( G^c \). Clearly it is a linear net, since it may be looked upon as ruling out certain of the decisions possible in \( G \). \( G^r \) is not uniquely determined by the above definition, however, since more than one subsets of \( \mathcal{R}(G) \) may satisfy the defining conditions. Also, any vertex of \( G \) will clearly appear on at least one of the two subgraphs \( G^c \) and \( G^r \) of \( G \).

We separate \( G \) into \( G^c \) and \( G^r \) in order that \( G^c \) may be transformed independently of \( G^r \) into a top-down subgraph (sub-step
3.1 in Alg. 5.4.1) and then replaced in the net (sub-step 3.4). This enables us to employ a type#2 reduction with \( G^c \) playing the role of \( G^* \).

The rest of this Section is presented in two parts: Part A below presents the algorithm after defining some related notation, part B following that contains the proof of correctness of the algorithm.

A. Let \( G^c, G^r, x^c_E \) etc. be as defined.

i. For any \( x_i \in x^c_E \) denote the subgraph of \( G^r \) in the neighbourhood of \( x_i \) (ref. Def. 4.2.1) by \( SN^r(x_i) \); and denote the corresponding set of last transitions in \( G^r \) by \( T^r_{\text{last}}(x_i) \).

**Ex. 5.4.3** For the net of the previous example, we show below the subgraphs \( SN^r(x_1) \) and \( SN^r(x_2) \) on the left and right respectively:

```
  x_i -> o -> | t_2
  \ |
  \ |
  \ |
  t_2
```

Here \( T^r_{\text{last}}(x_1) = T^r_{\text{last}}(x_2) = \{ t_2 \} \).

ii. As part of Alg. 5.4.1 we shall transform \( G^c \) into a top-down subgraph satisfying the following:

condition 5.4.1 \{ \forall R \in R(G^c), \forall x \in x^r_E \}

\[ C(R, x) = \text{set of all blank cycles in } R \]

It can easily be shown that this condition ensures condition 5.1.2, which is necessary for a type#2 reduction of \( G^c \).

iii. For the net \( G^c \), assume that it satisfies: \( FC(G^c) \subseteq \bigcup_{R \in R(G^c)} x^r_0 \).

(This is ensured by step 1 in Alg. 5.4.1.) Under this assumption define the set \( T_{pl} \subseteq T^{-1}(G^c) \) as follows:
\( T_{pl} = \left\{ t \in T^{-1}(G^C) \mid \sim \left[ \exists R \in \mathcal{R}(G^C) \text{ s.t. an } x \in X^R_e \text{ is on a cycle in } R \right] \right\} \)

**Ex. 5.4.4** In the graph of the previous example \( T_{pl} = \{ t \} \) because neither \( x_1 \) nor \( x_2 \) is on a cycle (in any of the three non-basic Traces) containing \( t \).

**T** \( _{pl} \) is simply the set of transitions of \( G^C \) which are incident on pipe-line branches in \( G^C \). To be able to reduce \( G^C \) by means of a type#2 reduction, we shall absorb the transitions in \( T_{pl} \) into the other branches of \( G^C \) by applying to them the transformation A2 of Appendix II (step 2 in Alg. 5.4.1).

We now have all the definitions needed, and present below Alg. 5.4.1. Note that step 3 processes \( G^C \) and \( G^R \) independently, and links the two together at sub-step 3.4. (Sub-steps 3.1 & 3.2 apply to \( G^C \), while sub-step 3.3 applies to \( G^R \).) Sub-steps 3.5 - 3.7 consist of type#2 reduction, replacement, etc.

**Alg. 5.4.1** input : non-basic linear FCP net \( G \) in FPP \( F \) cluster of innermost Traces \( G^C \) in \( G \) output : top-down FPP \( F' \) simulating \( F \)

step 0. Find the remainder net \( G^R \) (ref. Def. 5.4.2)

Let \( X = FC(G^C) - \bigcup_{R \in \mathcal{R}(G^C)} X^R \). 

step 1. If \( X = \emptyset \) then go to step 2

Otherwise carry out the following sub-steps:

1.1 Choose an arbitrary \( x \in X \) satisfying \( FC(G) \cap P^{-1}[^{SN}(x)] \) 

\[ = \{ x \} \text{ (ref. Section 4.2)} \]

1.2 Apply transformations \( T_1, T_2, T_3 \) to \( SN(x) \)

1.3 Apply Alg. 5.2.2 to \( SN(x) \), i.e. transform it to a top-down net simulating the original
1.4 Reduce $\text{SN}(x)^{**}$ to a transition $t^*$, and the corresponding operator $f^*$, i.e. a type#1 reduction of Section 5.1

1.5 Apply Alg.5.2.2 to the reduced net

1.6 Replace every occurrence of $f^*$ in the resulting net by $\text{SN}(x)$

1.7 Go to step 4.

step 2. Apply transformation A2 of Appendix II to $G^C$ and $T_{pl}'$, eliminating the pipe-line branches of $T_{pl}$ from $G^C$.

step 3. 3.1 Using an arbitrary $x \in X_E^C$ as the initial and terminal vertex, apply Alg.5.2.2 to $G^C$; close the initial-terminal link in the resulting net**

3.2 Ensure condition 5.4.1 through an appropriate application of transformation A2 to $G^C$

3.3 Apply transformation A3 of Appendix III to $G^R$, for each subgraph $\text{SN}(x_i)$, $x_i \in X_E^C$ **

3.4 Link the FC places in $\text{FC}(G^C) \cap \text{FC}(G^R)$ by adding the required output transitions from $G^C$ to $G^R$ and vice versa$^\$$

3.5 Reduce $G^C$ to a transition $t^*$ and FC place $x^*$, and the associated operator $f^*$ and predicate $g^*$, by means of a type#2 reduction

3.6 Apply Alg.5.2.2 to the reduced net

3.7 Replace every occurrence of $f^*, g^*$ in the resulting net by $G^C$.

step 4. The net obtained, and the corresponding FPP, is the desired output.

** We continue to denote by $\text{SN}(x), G^C$, and $G^R$, respectively, the nets found at the end of sub-steps 1.3, 3.1, and 3.3.

$$^\$$ This step is described more fully below.
Note the type#1 reduction/replacement in step 1, and the type#2 reduction/replacement in step 3. They are applied after appropriate preparatory transformations are made in steps 1.2, 2, 3.1 - 3.4, etc.

Sub-step 3.4 consists in linking the nets $G^C$ and $G^R$, and is carried out on each $x \in FC(G^C)$ satisfying the following condition 5.4.2

$$x \in FC(G^C) \cap FC(G^R) \text{ and } \left[ x' \neq x' \cap T^{-1}(G^C) \text{ or } x' \neq x' \cap T^{-1}(G^R) \right]$$

The corresponding output transitions of $x$ in $G^C$ and $G^R$ are brought together as diagrammed below, where we have assumed, as an example, that $x' \cap T^{-1}(G^C) = \{t_1\}$ and $x' \cap T^{-1}(G^R) = \{t_2\}$.

Superscripts $c$ and $r$ above refer to nets $G^C$ and $G^R$ respectively. The transitions $t^R_1$ and $t^C_2$ are added at sub-step 3.4 for the FC place $x$ of this example.

In part B below, we shall now present a proof of correctness of Alg.5.4.1.

B. That the various sub-steps of step 1 constitute valid transformations follows from the arguments of Sections 4.2, 5.1 and 5.3, since each of these sub-steps is an application of transformations described in these Sections. That step 2 is valid follows from the fact that it is an application of transformation A2 of Appendix II to $G^C$. Therefore, to prove the correctness of Alg.5.4.1 we shall prove the following:
i. sub-steps 3.1 - 3.4 constitute a valid transformation on G,

and ii. the net resulting from sub-steps 3.1 - 3.4 satisfies the necessary conditions for the type#2 reduction/replacement of sub-steps 3.5 - 3.7.

The above two statements are proved below in I and II respectively. Note that the validity of sub-steps 3.5 - 3.7 will then follow from the arguments of Section 5.1.

I. First we show that the net resulting from sub-steps 3.1 - 3.4 is a linear FCP net; we shall use here the SM-reduction criterion as before.

(a) Consider the net $G^r$ at the end of sub-step 3.3. We shall first show that it is a linear net.

Clearly sub-step 3.3 will modify at most those SCSM-components of $G^r$ which contain an element of $X^c_E$. Therefore let $S$ be an SCSM-component of $G^r$ containing an $x \in X^c_E$. We then know from Section 4.2 that $S$ contains exactly one element of $T^r_{\text{last}}(x)$, say $y$, and that $T^r_{\text{last}}(x) \supset T_{\text{last}}(x)$. Also, if $y \in T_{\text{last}}(x)$ then $S$ remains unmodified. Otherwise, i.e. if $y \in T^r_{\text{last}}(x) - T_{\text{last}}(x)$, then $S$ has the form

Then the edge $(z,y)$ above is deleted during transformation A3 (applied at sub-step 3.3, see Appendix III and the example given). Also, an edge $(z,u)$ is added, for an arbitrary $u \in T_{\text{last}}(x)$. Consider an SCSM-component, say $S'$, of $G^r$ which contains $u$
and y. The SM-allocations corresponding to S and S' will then differ in value only at y. A3 then modifies S s.t. S also contains u and y, and is a valid SCSM-component of the resulting net G^r.

Therefore corresponding to any component of the original net, we can find a component of the net resulting from sub-step 3.3, and the net is a linear FCP net.

(b) Now consider the net resulting from sub-step 3.4. As before, let S be an SCSM-component of G^r at the end of sub-step 3.3 which contains an x ∈ X_C. Sub-step 3.3 ensures that \( \forall x_i, x_j \in X_C, T_{\text{last}}(x_i) = T_{\text{last}}(x_j) \). Therefore S contains each element of X_C.

At sub-step 3.4, the SM-allocations of G^c and G^r are independent of each other. Also, in view of condition 5.4.1, the SCSM-components of G^c are blank. Therefore any SCSM-component S' of G^c will combine with S at sub-step 3.4 as diagrammed below:

Here we have shown an arbitrary \( x \in FC(G^c) \cap FC(G^r) \), with \( \{ t_1 \} = x \cdot \cap T^{-1}(G^c) \) and \( \{ t_2 \} = x \cdot \cap T^{-1}(G^r) \).

But then the above subgraph will form a valid SCSM-component of the resulting net, since S and S' are valid components in G^r and G^c respectively. By the SM-reduction criterion, we have that the net resulting from sub-step 3.4 is therefore a linear FCP net.

(c) We must now show that the output net of sub-step 3.4
will yield an FPP simulating the FPP \( F \), when the appropriate operators, predicates, etc. are assigned to the vertices of that net.

Let \( \alpha \) be an arbitrary firing sequence of \( G \). Then \( \alpha \) has a number of occurrences of transitions in \( X_E^C \cdot T^{-1}(G^r) \), each of which, except possibly the last, is followed by one or more occurrences of transitions in \( X_E^C \cdot T^{-1}(G^c) \). Firing sequences of \( G^c \), followed by those of the appropriate \( SN^r(x) \), occur as subsequences of \( \alpha \).

We also know that Alg. 5.3.1 is applied to \( G^c \) in order to find a top-down net simulating it, and that therefore a sufficiently large subset of the firing sequences of \( G^c \) is maintained at sub-step 3.4. Similarly, A3 maintains a sufficiently large subset of the firing sequences of \( SN^r(x) \), for each \( x \in X_E^c \). It follows that for each of the firing sequences of \( G^c \) and \( SN^r(x) \) occurring as subsequences of \( \alpha \), the output net has at least one sequence s.t. the two respective behaviour sequences will be related by the relation \( \text{sim} \). (ref. Section 2.3). Since we know (from condition 5.4.1 and Section 4.3) that \( G^c \) is live if any \( x \in X_E^c \) is marked, it follows that the net at the end of sub-step 3.4 has at least one firing sequence \( \alpha' \) s.t. the behaviour sequences corresponding to \( \alpha \) and \( \alpha' \) are related by the relation \( \text{sim} \). Since \( \alpha \) was chosen to be an arbitrary firing sequence of \( G \), it follows that the net at the end of 3.4 will yield an FPP simulating \( F \), with the appropriate interpretation.

Parts (b) and (c) together show that sub-steps 3.1 - 3.4 constitute a valid transformation on \( G \). This completes part I of the present proof.
II. We must now show that at the end of sub-step 3.4, \( G^c \) satisfies the necessary conditions for the type#2 reduction and replacement of the following sub-steps.

Condition 5.4.1 on \( G^c \) ensures that \( \forall R \in R(G^c), T_R = T^{-1}(\overline{R}_I) \) since it ensures that every transition on \( R \) is on a blank cycle in \( R \). Also, we have seen in (b) above that any SCSM-component \( S \) of \( G^F \) contains either none or all elements of \( X^c_E \) at the end of sub-step 3.3. In view of the fact that \( G^c \) satisfies condition 5.4.1 it follows that any SCSM-component of the net at the end of sub-step 3.4 contains either all or no elements of the set of output and input vertices of \( G^c \). Therefore \( G^c \) satisfies both parts of condition 5.1.2 applying to subgraphs of type#2.

Part II is therefore complete.

I and II taken together complete the proof of validity of Alg.5.4.1. In the following Section we present an example of the complete algorithm of Sections 5.1 - 5.4 applied to a linear net \( G \).

5.5 Illustrative Example

Consider the following linear FCP net \( G \), where the only cluster of innermost Traces \( G^c \) is as shown in bold lines. Then
$X_1 = \{y_1, y_2\}$ and $X_2 = \{x_1\}$. Also, note that $T_{\text{last}}(x_1) = \{t_{13}, t_{14}\}$ while $T_{\text{last}}(x_1) = \{t_2, t_{13}\}$. The net $G^r$ in this case is identical to the first net shown in Ex.5.3.1.

Then the top-down net obtained on applying Alg.5.2.2 to the above net is shown below. This net may be compared with the second net of Ex.5.3.1, so that the effect of the extra vertices in the net of this Section may be seen.

The transitions $t_1^{15}$ and $t_2^c$ in the above graph are introduced at sub-step 3.4 of Alg.5.4.1. The place $q$ is moved to the input of $t_{13}$ during transformation A3 applied to $x_1$.

The description of the structuring algorithm is now complete. The intuitive result that any FPP can be simulated by a top-down FPP has therefore been formally established. Also, we shall see that the algorithms of this Section will be useful in proving one part of the result of the following Chapter, where we shall determine the conditions under which an FPP can be simulated by a top-down FPP without any loss of parallelism involved. In other words, we shall determine conditions under which an FPP has a top-down equivalent (ref. Section 2.3 and Chapter 3).
CHAPTER 6

LOSS OF PARALLELISM

For the class of parallel programs modeled in this paper, the main result of the previous Chapter was that any parallel program may be simulated by a top-down parallel program. We mentioned in passing that in certain cases the top-down program involves the addition of sequencing constraints between operators which are in parallel in the original program. In terms of our definition of 'simulates' (Chapter 2), this means that, if \( F \) and \( F' \) denote the original and the top-down programs respectively, then the set of behaviour sequences of \( F' \) is smaller than that of \( F \). Or, intuitively, the degree of parallelism attainable in the top-down program is smaller than that in the original program.

In the present Chapter we shall examine further this phenomenon of loss of parallelism. In particular, we shall determine necessary and sufficient conditions for an FPP to have a TDP equivalent (ref. Chapters 2 & 3). It will be seen that the basic cause of this loss of parallelism is one type of a 'knotty' combination of parallel sequences at two alternate outputs of a decision node. (Recall that in an FCP net the decision nodes are the Free Choice places.)

Section 6.1 presents some basic definitions which are first needed before the conditions are described in Section 6.2. In Sections 6.3 and 6.4 we shall show that the conditions of Section 6.2 are necessary and sufficient. The analysis of Sections 6.1 - 6.3 is completely independent of Chapter 5, while Section 6.4 is, necessarily, highly dependent on Chapter 5.
6.1 Basic Definitions

For the purpose of analysis of this Chapter, we let \( \mathcal{F} = (G, S_{op}, f_{op}, S_{pr}, f_{pr}) \) be an FPP satisfying the following:

(i) \( f_{op} \) and \( f_{pr} \) are one-to-one functions, and

(ii) \( \lambda \notin f_{op}(T) \), where \( G = (T, P, E) \) as usual.

We then let \( \mathcal{F}' = (G', S_{op}, f_{op}', S_{pr}, f_{pr}') \) be a TDP simulating \( \mathcal{F} \).

The reason behind restrictions (i) and (ii) is this: whether or not \( \mathcal{F} \) has a TDP equivalent is determined by two factors: the net \( G \), and the interpretation on \( G \). We are here studying only the first one of these factors as it determines the existence of a top-down equivalent to \( \mathcal{F}' \). By restricting the interpretation as in (i) and (ii) above, we ensure that if \( \mathcal{F} \) has a top-down equivalent, then any other FPP with net \( G \) has a top-down equivalent.

The definitions that follow provide us with the basic ideas in terms of which the conditions of Section 6.2 are formulated. We shall use the naming convention described in Chapter 2, and we shall denote as a non-null transition any transition of \( G' \) which is not a \( \lambda \)-transition.

**Def. 6.1.1** Let \( p \in P_R \) for a Trace \( R = (T_R, P_R, E_R) \) of \( G \). We say \( p \) is a constraint in \( R \) between \( t_2 \) and \( t_1 \) if \( \{t_1\} = p \cap T_R, \{t_2\} = p \cap T_R, t_1 \notin \{p\} \) and \( t_2 \notin \{p\} \). We say \( p \) is a redundant constraint in \( R \) if \( t_1 \) and \( t_2 \) are on a 1-SCSM component \( S \) of \( G \) s.t. no element of \( t_1 \cap t_2 \) is on \( S \).

**Ex. 6.1.1** Consider the following net \( G \):

![Diagram of net G](image)
R is as shown in bold lines. Place p is a redundant constraint in R because the following 1-SCSM component of G contains \( t_1 \) and \( t_2 \) but not \( p \).

It can easily be verified that deleting \( p \), along with its input and output edges, from the above net will not cause any change in the set of firing sequences of the net. Transitions \( t_2, t_3, t_1 \) are constrained to fire in that order, and the place \( p \) is not required to ensure this constraint.

Def.6.1.2 Let \( p \) be a redundant constraint in Trace \( R=(T_R, P_R, E_R) \). We delete \( p \) in \( R \) by restricting \( R \) to transitions \( T_R \) and places \( P_R - \{ p \} \). We denote by \( R^t \) the marked graph obtained on deleting all the redundant constraints of \( R \), and say \( R^t \) is trimmed \( R \).

In view of the above two definitions, the following can be shown to follow from well-known properties of marked graphs.

Thm.6.1.1 (Property of marked graphs) Let \( M \) be a live and safe marking over \( G \) s.t. \( R \) is live and safe under \( M \). Let \( M^t \) be the restriction of \( M \) to \( R^t \). Then the set of firing sequences of \( R^t \) under \( M^t \) is identical to the set of firing sequences of \( R \) under \( M \).

The next definition is central to the conditions under which \( F' \) can be equivalent to \( F \). Ex.6.1.2 below provides some motivation behind the definition by attempting to show the need for it in a simple graph.

Ex.6.1.2 Consider \( F \) as shown below:
Let $R_1, R_2$ be Traces of $G$ corresponding to outputs $f_1, f_2$ of $p$. Then the following TDP $F'$ can be seen to be equivalent to $F$.

$F'$ is obtained from $F'$ by means of the following two steps:

(i) delete the place $q$, along with its input and output edges, since it is a redundant constraint in $R_1$ and in $R_2$,

(ii) make two disjoint copies of the subgraph of $G$ shown in bold lines — one in $R_1$ and one in $R_2$.

The definition below will lead us to the conditions under which the second step above can be carried out, in general, without introducing additional constraints in $G$.

**Def. 6.1.3** Let $R=(T_R, P_R, E_R)$ be a Trace of $G$ and let $p$ be an FC place in $R$. A **termination of $p$ in $R$** is any subset of $P_R$ obtained by the following procedure:

Let $t$ be the output transition of $p$ in $R$, and let $M^+ = \{ x \in P_R \mid M^0(x) = 1 \}$.

**step 0.** $Q = t \cdot P^{-1}(R^t)$ is a potential termination of $p$ in $R$.

Also, the sets $Y$ and $\text{passed}(R, p, Q)$ are given as $\emptyset$ and $\{ t \}$ respectively.

**step 1.** If $Q'$ is a potential termination of $p$ in $R$, and $\exists t' \in T_R$ s.t. $[(t' \cdot M^+) \cap P^{-1}(R^t)] \subseteq Q'$ then $Q'' = [Q' - t'] \cup T_R$. 
\[(t'\cdot Y) \cap P^{-1}(R^t)\] is a potential termination of \(p\) in \(R\). The set \(Y\) is updated to \(Y = Y \cup (t' \cap M^+)\) and the set 
\(\text{passed}(R,p,Q')\) is computed as 
\(\text{passed}(R,p,Q') = \text{passed}(R,p,Q) \cup \{t'\} \cup Q'.\)

step 2. A potential termination \(Q'\) of \(p\) in \(R\) is a termination of \(p\) in \(R\) if 
\(\bigcup_{x \in Q'} C(R^t,x) = C(R^t,p).\) 

A termination of \(p\) in \(R\) is, intuitively, a 'front' of tokens as it may be observed in \(R\) if transitions are fired in \(R\) starting at \(p\), provided this 'front' satisfies the condition of step 2 above. The set \(Y\) is required in the computation of all the terminations of \(p\) in \(R\), while the set \(\text{passed}(R,p,Q)\) for a termination \(Q\) will be needed in the following Sections. The following example, and Lemma 6.1.1, will clarify the idea of terminations.

**Ex.6.1.3** Let linear FCP net \(G\) be as shown:

In the above net let \(R\) be the Trace containing \(t\) and let \(1,\ldots,11\) be places of \(R\) as numbered. Then \(\{1,2\},\{2,3\},\{2,5\},\{5,10\},\{11\}\) etc. are terminations of \(p\) in \(R\). The sets \(\{3,4\},\{5,6\},\{7,9,10\}\) etc. are potential terminations but not terminations of \(p\) in \(R\) because they do not satisfy the condition of step 2, in view of the cycle 4-6-7-8 in \(R\) which does not contain \(p\). The final value of \(Y\) for this example will be \(\{8\}\), and the value of \(\text{passed}(R,p,\{2,5\})\), for example, is \(\{t,t',t'',1,3\}\) — which is simply the set of vertices passed in obtaining \(\{2,5\}\) from \(p\). 

\[\]
Another way to look at a termination of place $p$ in Trace $R$ is provided by the following Lemma.

**Lemma 6.1.1** Let $p$ and $R$ be as defined above and let $M^p$ be the marking over $R^+_t$ given by:

$$\forall q \in P^{-1}(R^+_t), \quad M^p(q) = \begin{cases} M^0(q), & q \neq p \\ 1, & q = p \end{cases}$$

Let $\alpha$ be any firing sequence of $R^+_t$ under $M^p$ of non-zero length, and let $M = \delta(M^p, \alpha)$. If a subset $Q$ of marked places of $R^+_t$ under $M$ satisfies $\bigcup_{x \in Q} C(R^+_t, x) = C(R^+_t, p)$ then $Q$ is a termination of $p$ in $R$.

**Proof** The proof is by induction on $|\alpha|$, the length of $\alpha$. For $|\alpha| = 1$ the result follows at once from step 1 of Def.6.1.3. Otherwise, $\alpha = \alpha'.t'$, where $|\alpha'| = |\alpha| - 1$. If all places in $Q$ are marked under $\delta(M^p, \alpha')$ then the result follows from the induction hypothesis. Otherwise, $t' \cap Q \neq \emptyset$. But then $X = P^{-1}(R^+_t) \cap [(Q - t') \cup t']$ satisfies $\bigcup_{x \in X} C(R^+_t, x) = C(R^+_t, p)$ and therefore $X$ is a termination of $p$ in $R$, by the induction hypothesis. The result then follows from steps 1,2 of Def.6.1.3 and the premise that $\bigcup_{x \in Q} C(R^+_t, x) = C(R^+_t, p)$.

The above Lemma justifies the intuitive meaning of a termination as being a 'front' of tokens observed on firing the transitions of $R$ starting at $p$. In the net $G$ of the previous example, the set $\{2,5\}$ can be seen to be obtained on firing the transitions $t,t',t''$ — which are the only transitions in the set $\text{passed}(R,p,\{2,5\})$.

We shall now make another definition which will be needed to formulate our criterion for the existence of a TDP equivalent to $F$.
Def. 6.1.4 Let \( p \in FC(G) \) and let \( R_i = (T_{R_i}, P_{R_i}, E_{R_i}) \), \( i = 1, 2 \), be Traces of \( G \) s.t. \( p \in P_{R_1} \cap P_{R_2} \). \( R_1 \) and \( R_2 \) are said to differ at \( p \) if \( p' \cap T_{R_1} \neq p' \cap T_{R_2} \). \( R_1 \) and \( R_2 \) are said to differ only at \( p \) if they do not differ at any other place in \( P_{R_1} \cap P_{R_2} \).

Recall that a Trace of \( G \) is a subgraph of \( G \) which corresponds to firing transitions of \( G \) with a fixed decision pattern at the FC places, and that in our terminology such a decision pattern is specified by an MG-allocation of \( G \). The above definition states that \( R_1 \) and \( R_2 \) differ in their respective decision patterns at exactly the one place \( p \). This fact is formally brought out in the following Lemma.

Lemma 6.1.2 Let \( R_1, R_2, p \) be as in the above definition s.t. \( R_1, R_2 \) differ only at \( p \). Then there exists an MG-allocation \( AM_1 \) of \( G \) s.t. the following are satisfied:

(i) \( AM_1 \) yields \( R_1 \) upon MG-reduction, and

(ii) \( AM_2 \), the MG-allocation of \( G \) obtained from \( AM_1 \) as shown below, yields \( R_2 \) upon MG-reduction:

\[
\forall x \in FC(G), \ AM_2(x) = \begin{cases} 
AM_1(x), & x \neq p \\
y, & x = p \text{ where } \{y\} = p' \cap T_{R_2}.
\end{cases}
\]

Proof Since \( R_1, R_2 \) differ only at \( p \), the following defines an MG-allocation \( AM_1 \) on \( G \):

\[
\forall x \in FC(G), \ AM_1(x) = \begin{cases} 
R_1(x), & x \in P_{R_1} \\
R_2(x), & x \in P_{R_1} - P_{R_2} \\
\text{arbitrary, otherwise},
\end{cases}
\]

where \( R_1(x) \) here denotes the output transition of \( x \) on \( R_1 \). We shall show that \( AM_1 \) as defined above satisfies the statement of the Lemma.
That $AM_1$ yields $R_1$ upon MG-reduction follows from the fact that each edge on $R_1$ is allocated, since $R_1$ is known to be a trace of $G$. That $AM_2$ obtained from $AM_1$ as specified will yield $R_2$ follows from the fact that changing the value of the allocation at $p$ from $R_1(p)$ to $R_2(p)$, using the notation defined above, results in every edge of $R_2$ being allocated.

Intuitively, the relation 'differ only' may be interpreted as specifying the way in which the 'switches' at the FC places are set. Then 'flicking the switch' at $p$ from $R_1(p)$ to $R_2(p)$ changes the resulting trace from $R_1$ to $R_2$, and vice versa. The following example illustrates this idea.

**Ex.6.1.4** Let $G$ be as shown below:

![Diagram](image)

The net has 4 traces, one corresponding to every possible allocation. Let $R_i,j$, $i=1,2$, $j=3,4$, denote the trace containing $t_i$ and $t_j$. Then, for example, $R_1,3$ and $R_1,4$ differ only at $p_2$. Similarly $R_2,3$ and $R_1,3$ differ only at $p_1$, while $R_1,3$ and $R_2,4$ are not thus related.

We close this Section with a Lemma which will be used in proving the necessity of the condition in the following Section. Def.6.1.5 below is required before we can present the Lemma.

**Def.6.1.5** Let $F$ and $F'$ be as defined, and let $R$ and $R'$ be traces of $G$ and $G'$ respectively. We say $R$ and $R'$ are equivalent if, for any marking $M$ reachable in $G$ which is live and safe over $R$, there exists a marking $M'$ reachable in $G'$ which is live and safe over $R'$ and satisfies the following:
Let $FS(R, M)$ and $FS(R', M')$ denote the firing sequences of $R$ and $R'$ under markings $M$ and $M'$ respectively. Then $\beta[FS(R, M)] = \beta[FS(R', M')]$, where $\beta$ is as defined in Section 2.2.

**Lemma 6.1.3** If $\mathcal{F}'$ is equivalent to $\mathcal{F}$, then for any Trace $R$ of $G$ there exists a Trace $R'$ of $G'$ s.t. $R$ and $R'$ are equivalent.

**Proof** Let $M$ be any marking reachable in $G$ which is live and safe over $R$. Then the set $\beta[FS(R, M)]$, using the notation of Def. 6.1.5 above, contains behavior sequences of unbounded length if $R$ does not contain 'start'. Then, since $G'$ is a finite net and $\mathcal{F}'$ is equivalent to $\mathcal{F}$, it follows that the fixed decision pattern corresponding to $R$ in $G$ will yield a Trace $R'$ of $G'$ satisfying the statement of the Lemma. If $R$ contains 'start', on the other hand, the result follows from the fact that firing the transitions of $R$ yields a marking under which no other transitions of $G$ can be made firable.

The proof of the above Lemma may be considered as an application of (10) to the case of two equivalent FPPs. Each Trace of an FCP net is a marked graph, and if $\mathcal{F}'$ is equivalent to $\mathcal{F}$, then for any Trace of $G$ there must be an equivalent Trace in $G'$.

This concludes the basic definitions etc. needed before we present the conditions under which $\mathcal{F}'$ can be equivalent to $\mathcal{F}$. Section 6.2 below presents the conditions, while Sections 6.3 and 6.4 present the appropriate proofs of their correctness.

### 6.2 Conditions for the Existence of TDP Equivalents

In terms of the basic definitions of the previous Section we shall now state the conditions under which an FPP has a TDP
equivalent. Using the notation of the previous Section, we say we are presenting the conditions that \( G \) must satisfy if \( F' \) is equivalent to \( F \).

The net of Ex. 6.2.1 will serve as a running example through this Section, and we first examine it in some detail.

**Ex. 6.2.1** Let the net \( G \) be as shown below:

\[ \text{Diagram} \]

\( G \) has 4 Traces. Two of the Traces, \( R_3 \) and \( R_4 \), do not contain 'start' and are shown in bold lines. Of the other two, let \( R_1 \) and \( R_2 \) correspond to outputs \( t_1 \) and \( t_2 \) of \( p \), respectively. Then the following net shows the Traces \( R_1 \) and \( R_2 \) of \( G \).

\[ \text{Diagram} \]

We shall now define some notation which we shall illustrate with the above net. The notation will then be used in formulating the condition we are looking for.

**I.** Let \( U \) denote the transitions of \( G \) which appear only on basic Traces (i.e. Traces containing 'start') of \( G \). In other words, \( U = \left\{ t \in T \mid \forall R \in \mathcal{R}(G), t \in T^{-1}(R) \Rightarrow 'start' \in P^{-1}(R) \right\} \).

Note that as an immediate consequence of the definition, any \( t \in U \) will appear at most once in a firing sequence of \( G \).
For the net of Ex. 6.2.1, the transitions \( t_1, t_2, t_{11}, t_{12}, t_{13}, t_{14} \) etc. are in \( U \). Transition \( t_{10} \) is not in \( U \) because it is on \( R_4 \) which does not contain 'start'.

II. Now let \( R=(T^R_R,T^R_P,T^R_E) \) be a Trace of \( G \) and let \( p \in P^R \). We denote by \( Z_b(R,p) \) the set

\[
\left\{ t \in T^R_R \mid \exists t' \text{ in } T^R_R \text{ s.t. } t' \text{ is in parallel with } p \text{ and in sequence with } t \right\} - U
\]

where we say vertices \( x,y \) of \( G \) are in sequence if a 1-SCSM-component \( S \) of \( G \) contains \( x \) and \( y \), and in parallel otherwise.

Consider the Trace \( R_3 \) of the net of Ex. 6.2.1 and the place \( q_2 \) on it. Transitions \( t_3, t_4, t_5 \) satisfy the condition that \( t_6 \) is in sequence with each of them but in parallel with \( q_2 \). Therefore \( Z_b(R_3,q_2) = \{ t_3, t_4, t_5 \} \).

Now consider Trace \( R_1 \) and the place \( q_1 \) on it. \( t_7, t_8, t_{11}, \) and \( t_{13} \) satisfy the condition that \( t_{10}, t_{12} \) are in sequence with them but in parallel with \( q_1 \). However \( t_{11}, t_{13} \) are in \( U \), therefore \( Z_b(R_1,q_1) = \{ t_7, t_8 \} \).

III. Now let the set \( V \) be defined as follows

\[
V = \left\{ x \in X^R_e \mid R \text{ is a Trace of } G \text{ and } \exists t \in T^{-1}(R) \text{ s.t. } t \text{ is in parallel with } x \& x' \cap U = \emptyset \right\}
\]

Recall that \( X^R_e \) is the set of fundamental exits of \( R \) (ref. Section 4.3). For instance, in the net of Ex. 6.2.1, \( X^R_3 \) and \( X^R_4 \) are given by \( \{ p_3 \} \) and \( \{ p_2 \} \) respectively, while \( X^R_1 = X^R_2 = \{ 'end' \} \).

Now note that no transition in \( R_3 \) is in parallel with \( p_3 \), and no transition in \( R_4 \) is in parallel with \( p_2 \). However, \( t_{10} \) and \( t_{12} \) are in parallel with 'end' in \( R_1 \) (and \( R_2 \)). Therefore for the net of Ex. 6.2.1 we have \( V = \{ 'end' \} \).

IV. Finally let \( Z_h \) be the set defined as follows:
\[ Z_h = \left\{ x \in FC(G) \mid \exists R, R' \in R(G) \text{ s.t. } x \in X^R_e, (R, R') \right\} \]

Intuitively, \( V \) is the set of fundamental exits from pipeline Traces. The defining condition for \( V \) is also the defining condition for the presence of pipeline branches within a Trace (ref. Section 4.3). \( Z_h \) then takes \( V \) (note that \( V \subseteq Z_h \)) and augments it by the fundamental exits of those Traces which are \( \sigma_{SN} \)-related to pipeline Traces. (Recall \( \sigma_{SN} \) is a partial order.)

We give below the Hasse diagram of \( \sigma_{SN} \) for the net of the first example of this Section.

\[ \begin{array}{cccc}
R_1 & R_2 \\
& R_3 & R_4
\end{array} \]

Note that this diagram is the formal equivalent of the statements: \( R_3 \) is a Trace in one of the parallel branches of \( R_1 \), etc. The motivation behind finding \( Z_h \) is, roughly speaking, this: if \( R' \) is pipeline, and \((R, R') \in \sigma_{SN}\), then in the present context \( R \) also behaves like a pipeline Trace.

We see below that the sets \( Z_b(R, p) \) and \( Z_h \) are used in characterizing the conditions under which \( F' \) is equivalent to \( F \). We now have all the definitions and notation — which are required for the purpose, and state below the conditions. The appropriate proofs etc. are in the following Sections.

**Condition 6.2.1**

(For the existence of a TDP equivalent to an FPP)

\( \forall p \in FC(G) \), and

\( \forall R_1, R_2 \in R(G) \text{ s.t. } R_1, R_2 \text{ differ only at } p, \)

there exists a termination \( \langle p, R_1 \rangle \) of \( p \) in \( R_1 \), for an
an $i \in \{1, 2\}$, satisfying the following:

(i) $\forall t \in \langle p, R_i \rangle \cap T_{R_j}, \forall t' \in \langle p, R_i \rangle \cap T_{R_j}, j = 1, 2, \left[t \cap t'\right] \neq \emptyset$

and (ii) $\text{passed}(R_1, p, \langle p, R_i \rangle) \cap Z_k = \emptyset$, for $k = 1, 2, 3$

where $Z_1 = Z_h \cap \left[ P_{R_1} \cap P_{R_2} \right]$

$Z_2 = Z_b(R_1, p) \cap Z_b(R_2, p)$

and $Z_3 = x_{e1}^{R_1} \cap x_{e2}^{R_2}$.

Condition (i) above states that input and output sets of transitions of $\langle p, R_i \rangle$ be pairwise mutually constrained. It will be seen that (i) ensures that the net can be modified, so as to satisfy the requirements of top-down nets, without the addition of any further constraints to it.

Condition (ii) above states that in obtaining $\langle p, R_i \rangle$ none of the vertices of $Z_k$, $k = 1, 2, 3$, should be passed. It will be seen that replication, which is a central step in all 'Structure Algorithms', is valid in parallel systems only if no pipe-line branches are being replicated. Condition (ii) is the formal statement of this requirement.

The net of Ex.6.2.1 satisfies condition 6.2.1. The set $\{q_3, q_4\}$ for example, is a valid termination of $p_1$ in $R_1$ and $R_2$. Similarly the set $\{p_3\}$ satisfies the condition for the place $p_3$ and the pair of Traces $R_1, R_3$ (and the pair $R_2, R_3$). We show below a top-down equivalent of the net. The operators $f_3, f_7, f_8, f_{10}$ (corresponding to $t_3, t_7, t_8, t_{10}$ respectively) occur twice in the net below and are represented by $t_{13}^{1}, t_{13}^{2}, \ldots, t_{10}^{1}, t_{10}^{2}$ respectively. To a large extent the relationship between the two nets, and the approximate method for obtaining the top-down net, are apparent in the following diagram.
Roughly speaking, the following two steps are used in obtaining the above net from that of Ex. 6.2.1.

(a) places \( q_3 \) and \( q_4 \) are brought together (at \( q_5 \)), and

(b) \( R_3 \) and \( R_4 \) are separated from the rest of the net, so that they have only the places \( p_3 \) and \( p_2 \), respectively, in common.

Condition 6.2.1 essentially states the restrictions that \( G \) must satisfy so that steps (a) and (b) may be carried out in general without adding any extra constraints to the net.

6.3 Necessity

In this Section we shall show that condition 6.2.1 of the previous Section is necessary if \( \mathbf{F}' \) is to be equivalent to \( \mathbf{F} \). To that end, we shall choose an arbitrary pair \( R_1, R_2 \) of Traces of \( G \) and an FC place \( p \) s.t. \( R_1, R_2 \) differ only at \( p \). We shall show then that the statement of condition 6.2.1 is implied by the premise : \( \mathbf{F}' \) is equivalent to \( \mathbf{F} \).

Therefore let \( R_1, R_2 \in \mathcal{R}(G) \) and \( p \in \mathcal{FC}(G) \) s.t. \( R_1, R_2 \) differ
only at $p$. Let $f_{pr}(p) = g_1$ (recall the naming convention, and
the assumption that $f_{op}, f_{pr}$ are one-to-one). Also, let $f_1, f_2$
be the output transitions of $g_1$ on $R_1, R_2$ respectively.

**Notation** Recall Def. 3.1 of proper marked graphs. In this Section
we shall represent a pmg schematically as shown below:

```
initial transition | terminal transition
```

The arguments of this Section then follow one of three
cases:

**case I.** Neither $R_1$ nor $R_2$ contains 'start'

**case II.** Both $R_1$ and $R_2$ contain 'start'

**case III.** (W.l.g.) $R_1$ contains 'start', $R_2$ does not.

As we shall see, the arguments follow a similar line in
each of the cases below. The argument is elaborated fully for
case I(a), and for the other cases we briefly outline how the
argument must be modified appropriately to fit each case.

**case I.** Neither $R_1$ nor $R_2$ contains 'start'

Now from Lemma 6.1.3 it follows that $\exists R'_1 \in R(G')$ s.t. $R'_1$
is equivalent to $R_1$. Therefore an FC place $g_1^{k_1}$, for some $k_1 \in$
$
\{0, 1, \ldots, |f_{pr}^{-1}(g_1)| - 1\}$, and a transition $f_1^{l_1} \in g_1^{k_1}$, for some

$l_1 \in \{0, 1, \ldots, |f_{op}^{-1}(f_1)| - 1\}$, appear on $R'_1$.

Since $G'$ is a top-down net, the Trace $R'_1$ has the form:

```
  \[ Z_1 \rightarrow Z_2 \rightarrow \ldots \rightarrow Z_k \rightarrow Z_{k+1} \rightarrow \ldots \rightarrow Z_n \]
```

```
  \[ Z_1 \rightarrow Z_2 \rightarrow \ldots \rightarrow Z_k \rightarrow Z_{k+1} \rightarrow \ldots \rightarrow Z_n \]
```
In other words, $R'_1$ is in the form of a number of pmsg nets (or single transitions) strung together by the places $z_1, \ldots, z_n$. Let the places $z_1, \ldots, z_n$ be called the SM-nodes of $R'_1$. (Each SM-node of $R'_1$ is a node of $R'_1$, ref. Def. 4.3.3.)

**Case I(a)** $g^k_{11}$ is not an SM-node of $R'_1$

In other words, $g^k_{11}$ is inside one of the pmsg nets in the above diagram. Now since $G'$ is a linear net, at least one node of $R'_1$ is an FC place. Let $g^k_{22}$ be such an FC place and let $f^1_{33}$ be the output transition of $g^k_{22}$ on $R'_1$. Then $R'_1$ can alternatively be drawn as shown below, by grouping the appropriate nodes together in the above diagram:

Now since $G'$ is a top-down net, every $x \in FC(G')$ is first introduced into it through an S-substitution step. Consider the psm $S$ substituted into $G'$ which contains the FC place $g^k_{11}$. Starting at $g^k_{11}$ we can construct a pair of paths $\tau_1$ and $\tau_2$ in $S$ corresponding to the decision patterns of $R_1$ and $R_2$ respectively (which differ only at $g_1$). The paths $\tau_1$ and $\tau_2$ can have one of two forms in $S$, as shown below, where $l_2 \in \{0, 1, \ldots, \lfloor f_{op}^{-1}(f_2) \rfloor - 1\}$.

(a) \hspace{5cm} (b)

Depending on the form of these two paths in $S$, therefore,
one of the following two subgraphs exists in $G'$:

(a) ![Diagram](image)

(b) ![Diagram](image)

Each of the above subgraphs is the union of two Traces of $G'$: $R'_1$, and the one shown in bold lines, say $R'_2$.

**Lemma 6.3.1** $R'_2$ is equivalent to $R_2$.

**Proof** Consider a marking $M'$ of $G'$ which is live and safe over $R'_1$, and consider the behaviour sequences of $F'$ obtained by firing transitions in $R'_2$ only (starting at $M'$). These behaviour sequences will be of unbounded length. Since we have assumed $f_{op}, f_{pr}$ to be one-to-one, the only Trace of $G$ which can exhibit these behaviour sequences in $F$ is $R_2$. Since $F'$ is equivalent to $F$ (premise) it follows that $R'_2$ is equivalent to $R_2$.  

Note that this proof has followed an argument similar to that in Lemma 6.1.3. In essence we have shown that a pair of Traces $R'_1, R'_2$ can be found in $G'$ corresponding to the pair $R_1, R_2$ of $G$ in the case that $g^k_1$ is not an SM-node of $R'_1$.

Now consider the subgraph (b) above, which corresponds to the paths $\tau_1, \tau_2$ shown in diagram (b) previously. The FC-place $g^k_1$ is an SM-node of $R'_2$ in this subgraph. We can therefore, w.l.o.g., interchange the subscripts 1 and 2 and apply to this pair the case I(b) below, if the union of $R'_1, R'_2$ has the shape of (b) above. W.l.o.g. therefore assume that $R'_1, R'_2$ have together the form of (a) above.
Let transition $t$ be as in (a) above. Then (a) can be redrawn as shown below, under a marking of $G'$ under which $t$ is the only transition fireable:

Let $A_1, A_2, A_3$ be the three pmg nets shown above.

We now find three sets $T_1', T_2', T_3'$ of transitions in $A_1, A_2, A_3$ respectively, according to the following steps:

step 0. consider the set of all blank elementary paths from $f_1^1$ to $t'$ (resp. $f_2^2$ to $t''$, $t$ to $g_1^k$)

step 1. for each path $\Pi$ above, find the last (resp. last, first) non-null transition on $\Pi$ as follows:

\[ \text{path } \Pi \]

if $x$ is the last (resp. first) non-null transition on $\Pi$ then $x$ is the only non-null transition on the terminal subpath of $\Pi$ with initial vertex $x$ (resp. the initial subpath of $\Pi$ with terminal vertex $x$)

step 2. of the set, say $T''_1$ (resp. $T''_2$, $T''_3$) found at step 2, find the subset $T'_1$ (resp. $T'_2$, $T'_3$) as follows:

\[ T'_1 = \{ x \in T''_1 \mid x \text{ is the last non-null transition on every blank elementary path (of type defined in step 1) containing } x \} \]

$T'_2$ — obtained by changing $T''_1$ to $T''_2$ above

$T'_3$ — obtained by changing $T''_1$ to $T''_3$ and last to first above

The following example illustrates these definitions.
Ex. 6.3.1 Let $A_1$ be as shown below:

There are three blank elementary paths from $f_1^1t_1t_2t'$ — $f_1^1t_1t_2t'$, $f_1^1t_1t_2t'$ and $f_1^1t_3t_2t'$. Then $T_1^1 = \{ t_2, t_3 \}$ because $t_2$ is the last non-null transition on the first two of these paths, and $t_3$ is the last non-null transition on the third. But $t_3$ is not the last non-null transition on the second path listed, and therefore $t_3 \not\in T_1^1$, i.e. $T_1^1 = \{ t_2 \}$.

The motivation behind defining $T_1^1, T_2^1, T_3^1$ is this: Consider $A_1$ and $A_3$. On either side of $q$, in $A_1$ and $A_3$ respectively, we here find the set of non-null transitions nearest to $q$. Because of the presence of $q$, any pair of transitions in $T_1^1 \times T_3^1$ is constrained to fire strictly in sequence. In other words, no $u \in T_1^1, v \in T_3^1$ can be enabled in parallel. The following diagram illustrates this schematically:

Now since $R_1^1$ is equivalent to $R_1$, we shall show that the set of places in $R_1$ between $f_{op}(T_1^1)$ and $f_{op}(T_3^1)$ forms a termination of $g_1$ in $R$ which satisfies the requirement of condition 6.2.1 for the FC place $g_1$ and the pair of Traces $R_1, R_2$.

Note: The basic line of argument remains unchanged in cases I(b), II and III below. The difference is that the subgraphs corresponding to $R_1', R_2'$ etc. have a somewhat different form. In each of these cases we shall only outline how the present argument
must be modified, and we shall omit the full details.]

To continue with the present argument, note that $T_1, T'_2, T'_3$ are non-empty because $f_{11}^1, f_{21}^1, f_{31}^1$ respectively are non-null.
The following properties can then be easily deduced.

**Lemma 6.3.2** If $|T_1| > 1$ (resp. $|T'_2| > 1$, $|T'_3| > 1$) then the transitions in $T_1$ (resp. $T'_2, T'_3$) are mutually in parallel.

**Proof** Consider any $x, y \in T_1$. By definition they are on blank elementary paths $\pi_x$ and $\pi_y$ in $A_1$ from $f_{11}^1$ to $t'$. Assume that $x, y$ are not in parallel, then there is a cycle $\tau$ in $A_1$ containing $x$ and $y$, and the following paths must exist in $A_1$:

But then a blank elementary path $\pi$ can be constructed above from $f_{11}^1$ to $t'$ s.t. both $x$ and $y$ are on it. But then, from step 2 of the construction for $T_1$, at most one of $x, y$ can be in $T'_1$. This contradicts our premise that $x, y \in T_1$. Our assumption is wrong and the result follows.

Let $T_1 = f_{op}(T'_1)$, $T_2 = f_{op}(T'_2)$, $T_3 = f_{op}(T'_3)$ be sets of transitions of $G$. As mentioned above, we shall show that the places in $R_1$ between $T_1$ and $T_3$, i.e. the set $T'_1 \cap T'_3$, is finally to be shown to satsify the requirements of condition 6.2.1. Note that since $f_{op}, f_{pr}$ are one-to-one, we have $T_1 \subseteq T^{-1}(R_1)$, $T_2 \subseteq T^{-1}(R_2)$ and $T_3 \subseteq T^{-1}(R_1) \cap T^{-1}(R_2)$.

**Lemma 6.3.3** (i) $\forall x \in T_1, \forall y \in T_3, x \cdot n \cdot y \neq \emptyset$

(ii) $\forall x \in T_1, \forall y \in T_3, x \cdot n \cdot y \neq \emptyset$

**Proof** (i) Assume the opposite, i.e. let $x \in T_1, y \in T_3$ s.t. $x \cdot n \cdot y = \emptyset$. Now, $x$ and $y$ cannot be in parallel in $R_1$ because the
corresponding transitions in $T_1$ and $T_3$ are strictly in sequence in $R_1'$. Therefore $x,y$ are strictly in sequence in $R_1$. Since we have assumed $x' \cap y = \emptyset$, a path such as the one below exists in $R_1$ for some $m \geq 1$:

$$
\begin{array}{c}
x
\end{array} \longrightarrow \begin{array}{c}
z_1
\end{array} \longrightarrow \begin{array}{c}
z_2
\end{array} \longrightarrow \ldots \ldots \longrightarrow \begin{array}{c}
z_m
\end{array} \longrightarrow \begin{array}{c}
y
\end{array}
$$

Therefore no firing sequence of $G$ can have $xy$ occurring in it. But from the structure of $R_1'$ and $R_2'$ we know that firing sequences of $G'$ contain $xy$ occurring together. This contradicts the fact that $F'$ simulates $F$, i.e. that $B(F') \subseteq B(F)$. In view of this contradiction, our assumption is wrong and the result follows. (ii) follows a similar argument.

### Lemma 6.3.4
\[ \forall q \in T_1' \cap T_3', C(R_1^t, q) \subseteq C(R_1^t, p) \]

**Proof** Assume the opposite. Then some $u \in T_1$, $v \in T_3$ and $w \in T_1^{-1}(R_1)$ satisfy the condition that $w$ is in parallel with $f_1$ but in sequence with $u$ and $v$. But from the structure of $R_1'$ we observe that no 3-tuple of transitions in $T_1' \times T_3' \times T_1^{-1}(R_1')$ can satisfy the corresponding condition in $R_1'$. Since $F'$ simulates $F$, we have a contradiction and the result follows.

### Lemma 6.3.5
$T_1' \cap T_3$ is a termination of $g_1 = p$ in $R_1$.

**Proof** First we show that $\bigcup_{x \in W} C(R_1^t, x) = C(R_1^t, p)$ where $W = (T_1' \cap T_3') \cap p^{-1}(R_1^t)$. Lemma 6.3.4 shows one-way containment. To show the converse, assume the opposite. Then $\exists t \in T_1^{-1}(R_1)$ s.t. $t$ is in sequence with $p$ but in parallel with all transitions of $T_1$. But then, from the structure of $R_1'$ we see that no corresponding transition can exist in $R_1'$. Since $R_1'$ is equivalent to $R_1$, we have a contradiction and the above set equality follows.

Now consider a marking $M'$ of $G'$ s.t. $R_1'$ is live under $M'$.
and \( g_{1}^{k} \) is marked. Consider a firing sequence \( \alpha' \) of \( R_{1} \) which fires all the transitions of \( A_{1} \), but none of \( A_{3} \). For the corresponding firing sequence of \( R_{1} \), therefore, say \( \alpha \), all the transitions of \( T_{1} \) appear in \( \alpha \) but none of \( T_{3} \). Therefore places in \( T_{1} \cap T_{3} \) are marked under the marking reached on firing \( \alpha \). The result follows from the above set equality and Lemma 6.1.1.

Note that Lemmas 6.3.4 and 6.3.5 together show that \( T_{1} \cap T_{3} \) is a termination of \( g_{1} = p \) in \( R_{1} \) satisfying the first half of condition 6.2.1. To complete the present case I(a) we now must show that \( \text{passed}(R_{1}, p, T_{1} \cap T_{3}) \cap Z_{b}(R_{1}, p) \cap Z_{b}(R_{2}, p) = \emptyset \), for \( k = 1, 2, 3 \), where \( Z_{b} \) are as defined in condition 6.2.1. This is shown in the following paragraphs.

(1) To show that \( \text{passed}(R_{1}, p, T_{1} \cap T_{3}) \cap Z_{b}(R_{1}, p) \cap Z_{b}(R_{2}, p) = \emptyset \). Assume the opposite, i.e. let \( x \in \text{passed}(R_{1}, p, T_{1} \cap T_{3}) \cap Z_{b}(R_{1}, p) \cap Z_{b}(R_{2}, p) \). By definition then, \( \exists x, y \in T_{1}^{-1}(R_{1}) \) s.t. \( x, y \) are in sequence, \( p, y \) are in parallel, and \( x \in \text{passed}(R_{1}, p, T_{1} \cap T_{3}) \). But \( x \in \text{passed}(R_{1}, p, T_{1} \cap T_{3}) \) implies that \( \exists x'_{1} \) on \( A_{1} \) s.t. \( f_{\text{op}}(x'_{1}) = x \). This is so because \( x \in \text{passed}(R_{1}, p, T_{1} \cap T_{3}) \rightarrow \exists z \in T_{1} \) s.t. the following path exists in \( R_{1} \), and a corresponding path

\[
\begin{array}{c}
\circ \quad g_{1} \\
\quad \quad x \\
\quad \quad z
\end{array}
\]

therefore must exist in \( R_{1} \). [This may be seen by considering \( R_{1} \) and \( R_{1}^{'} \) as two equivalent marked graph based control schemes as in (10).] Therefore \( \exists y'_{1} \) on \( A_{1} \) s.t. \( y'_{1} \) is on a pipe-line branch in \( A_{1} \) and \( f_{\text{op}}(y'_{1}) = y \).

Now from Lemma 4.1.1 it can be shown that there is a cycle \( \tau \) in \( R_{1} \) and \( R_{2} \) containing \( x \) and \( y \) but not \( g_{1} \), in view of our premise that \( x \in \text{passed}(R_{1}, p, T_{1} \cap T_{3}) \cap Z_{b}(R_{1}, p) \cap Z_{b}(R_{2}, p) \). Since
$y'_1$ is not on $R'_2$ it follows that $\exists y'_2$ on $R'_2$ s.t. $f_{op}(y'_2) = y$. But then since $y'_1$ is on a pipe-line branch in $R'_1$ it follows that we can enable $y'_1, y'_2$ together (in parallel) and obtain a behaviour sequence of $f'$ of the type $...yy...$. But since the cycle $\mathcal{T}$ containing $x$ and $y$ rules out such a behaviour sequence from $f$, we have reached a contradiction. Our assumption is wrong, i.e. $\overline{\text{pass}}(R_1, p, T'_1 \cap T'_3) \cap Z_b(R_1, p) \cap Z_b(R_2, p) = \emptyset$.

(2) To show $\overline{\text{pass}}(R_1, p, T'_1 \cap T'_3) \cap Z_h \cap \left[ P^{-1}(R_1) \cap P^{-1}(R_2) \right] = \emptyset$. Assume the opposite, i.e. $\exists x \in \overline{\text{pass}}(R_1, p, T'_1 \cap T'_3) \cap Z_h \cap \left[ P^{-1}(R_1) \cap P^{-1}(R_2) \right]$. Then $x \in x_R$ for some $R_3 \in \mathcal{R}(G)$ s.t. either $R_3$ has one or more transitions on pipe-line branches, or $(R_3, R_4) \in \sigma_{SN}$ for some Trace $R_4$ in $G$ which has one or more transitions in pipe-line branches. In either event, $\exists y$ not on $R_1$ s.t. $y$ is on a pipe-line branch in $R_3$ (or $R_4$) and a marking $M^V$ of $G$ is live over $R_1$ and enables $y$.

Again, as in (1) above, there is an FC place $x'_1$ on $A'_1$ s.t. $f_{pr}(x'_1) = x$. Since $\overline{f'}$ is equivalent to $f'$, there is a Trace $R'$ in $G'$ (corresponding to either $R_3$ or $R_4$ in $G$) s.t. a marking $M'^V_i$ of $G'$ enables a transition $y'_1$ on $R'$, $f_{op}(y'_1) = y$, and $R'_1$ is live under under $M'^V_i$. As in the case of transition $y$ in $G$, then, $y'_1$ is not on $R'_1$.

Since $x'_1$ is not on $R'_2$, and $x \in P^{-1}(R_1) \cap P^{-1}(R_2)$, it follows that $\exists x'_2$ on $R'_2$ having the same properties as $x'_1$. In particular a Trace $R''$ exists in $G'$ s.t. $R''$ is equivalent to either $R_3$ or $R_4$, and $x'_2 \in x'_R$. Therefore we can find a marking live over $R'_2$ which enables a transition $y'_2$ not on $R'_2$ and $f_{op}(y'_2) = y$. But this implies that $y'_1, y'_2$ can be enabled in parallel in $G'$, giving a behaviour sequence of the type $...yy...$. This contradicts
the fact that $\mathcal{F}'$ simulates $\mathcal{F}$. Our assumption must be wrong
and the result follows.

(3) To show \[ \overline{\text{passed}}(R_1, p, T_1 \cap T_3) \cap [x_{e_1} \cap x_{e_2}] = \emptyset. \]
Assume the opposite, i.e. \( \exists x \in \overline{\text{passed}}(R_1, p, T_1 \cap T_3) \cap [x_{e_1} \cap x_{e_2}] \).
But then, from the argument of (1) above, it must be that \( \exists x_i \)
on \( A_1 \) s.t. \( x_i \in x_{e_1}^{R_1} \). But since \( e_1 \) is not an SM-node of \( R_1 \), we
have \( x_{e_1}^{R_1} \cap P^{-1}(A_1) = \emptyset \). In view of the contradiction, the result
follows.

This completes all three parts of the second half of
condition 6.2.1. The proof for case I(a) is therefore complete.
For cases I(b), II and III we shall outline how the above
arguments must be modified. Before that we give an example of
condition 6.2.1 being violated, and attempt an intuitive mean-
ing into the condition.

**Ex. 6.3.2** The following net has two Traces which do not contain
'start'; let \( R_1, R_2 \) be these two Traces corresponding to output
transitions \( t_1, t_2 \) of \( p \) respectively.

![Diagram](image)

We have shown three places \( q_1, q_2, q_3 \) in bold lines because
only two possible terminations of \( p \) in \( R_1 \), namely \( \{q_1, q_2\} \) and
\( \{q_2, q_3\} \) are contained in \( R_2 \) and can therefore be candidates to
satisfy part (i) of condition 6.2.1 for \( R_1, R_2 \) and \( p \). In fact,
however, none of these terminations satisfies the requirement,
and the net violates condition 6.2.1.
Now we show below the FCP net which simulates the above net and is top-down:

The place q in the above net is now a termination of p in R_1 (and R_2) which satisfies both parts of condition 6.2.1 for this top-down net. We see that the above net has sequencing constraints between t_5 and t_3', and between t_6 and t_4', for example, which are not present in the original. We have had to add these constraints so as to satisfy the first part of condition 6.2.1.

The arguments for case I(b) are now presented, followed by cases II and III.

Case I(b) e^k_1 is an SM-node of R'_1.

In this case p = e_1 must be a node of R'_1. Consider the set of vertices, say X, which are common to R_1 and R_2. Let Y denote the set of transitions in R'_1 which correspond to operators in X, and s.t. each y ∈ Y is the occurrence of the operator f'_o (y) nearest to e^k_1 in R'_1. Since e^k_1 is an SM-node of R'_1, the set Y can be uniquely determined.

Now clearly a Trace R'_2 equivalent to R_2 exists in G'. If one or more transitions of Y are on R'_2 then R'_1 and R'_2 are not disjoint, and therefore have the form shown below, and each of the arguments of case I(a) applies as before to the three subnets A_1, A_2, A_3 to give us the desired result.
Otherwise, i.e. if no transitions of Y are on \( R'_2 \), then, for \( Z_1, Z_2, Z_3 \) as in condition 6.2.1, the set \( \{ g_i \} \) is the required termination, in \( R_1 \), since \( X \cap Z_i = \emptyset \) follows as in case I(a) for \( i = 1,2 \). If \( X \cap Z_3 \neq \emptyset \), however, the set \( \{ z \} \) for some \( z \in Z_3 \) will be the required termination.

**Case II** Both \( R_1 \) and \( R'_2 \) contain 'start'.

In this case \( R'_1 \) and \( R'_2 \) have the form shown below, and each of the arguments of case I(a) applies. To show part (ii)

of condition 6.2.1, we make use of the defining conditions of \( Z_h \) and \( Z_b (R, p) \) which take into account the transitions of \( G \) appearing only on basic Traces (ref. the set \( U \) of Section 6.2).

**Case III** (W.l.g.) \( R_1 \) contains 'start', \( R'_2 \) does not.

In this case the two Traces have the following form in \( G' \), and arguments of case I(a) applied to \( A_2 \) and \( A_3 \) below yield that

a termination of \( p \) in \( R_2 \) has the required properties.

All the necessary arguments of this Section are therefore
complete, giving us the following:

**Thm. 6.3.1** Let $\mathcal{F}, \mathcal{F}'$ be as defined. If $\mathcal{F}'$ is equivalent to $\mathcal{F}$ then $G$ satisfies condition 6.2.1.

### 6.4 Sufficiency

The sufficiency of condition 6.2.1 for the existence of a TDP equivalent to a given FPP will be shown with reference to the algorithms of Chapter 5. We shall show that if the original FPP satisfies condition 6.2.1, then suitable modifications exist to the algorithms of Chapter 5 s.t. the resulting FPP preserves all the parallelism of the original. Parts A and B below describe the modifications required on Alg. 5.3.1 and Alg. 5.4.1 respectively, and part C completes the arguments of this Section.

Let $\mathcal{F} = (G, S_{op}, f_{op}, S_{pr}, f_{pr})$ be the given FPP satisfying condition 6.2.1.

**A.** Recall that Alg. 5.3.1 takes as input a basic FCP net, and finds a top-down net simulating it. Consider the following modifications to Alg. 5.3.1, which are valid if the given net $\overline{G}$ (ref. Alg. 5.3.1) satisfies condition 6.2.1.

#### #(i) Insert the following immediately after step 0:

step 0.5 delete all redundant constraints in $\overline{G}$ (i.e. constraints redundant in every Trace of $\overline{G}$)

#### #(ii) Let $p$ be as defined at step 1 of Alg. 5.3.1. Then, since $\overline{G}$ is a basic net satisfying condition 6.2.1, there exists a Trace $R$ of $\overline{G}$ and a termination $\langle p, R \rangle$ of $p$ in $R$ s.t. $\forall R' \in \mathcal{R}(\overline{G}), [R, R' \text{ differ only at } p] \Rightarrow [\langle p, R \rangle \text{ satisfies both the clauses of condition 6.2.1 for the pair } R, R']$. 
Then we alter steps 2-6 of Alg. 5.3.1 as follows. Every occurrence of SN(p) in these steps is replaced by SN'(p) which is obtained as described below.

In Def.4.2.1, at step 3, we add the condition "if t ∈ [p,R]", and denote the resulting subgraph by SN'(p). This intuitively corresponds to constructing the subgraph up to the set of transitions [p,R], but no further. Note that SN'(p) may or may not be a subgraph of SN(p), and that it has all the relevant properties of SN(p) (i.e. Lemmas 4.2.1-4.2.4 etc.).

(iii) At step 2 of Alg.5.3.1 we apply transformations T1,T2 to SN'(p) but not T3.

(iv) We delete step 3 of Alg.5.3.1 and replace it by the subtransformation described below.

Consider the following condition on SN'(p), where Tlast(p) is obtained on replacing SN(p) by SN'(p) in Def.4.2.2:

\[
\forall x \in P^{-1}[SN'(p)] - \{p\} \cdot Tlast(p), \forall y \in p',
\]

If |'x| > 1 then

condition 6.4.1

\[
\exists \text{there is an elementary path } \pi \text{ from } y \text{ to a vertex in } [p,R] \Rightarrow \exists \text{ every elementary path } \pi' \text{ from } y \text{ to a vertex in } [p,R] \text{ contains } x
\]

We claim that a subtransformation involving replication ensures the above condition on SN'(p) without introducing any additional constraints into G. Consider the following example. Ex.6.4.1 The net G below satisfies condition 6.2.1 but not condition 6.4.1:
Now replication of \( f_3', f_4', f_5' \) as shown below causes condition 6.4.1 to be satisfied:

We do not describe the sub-transformation here in full, since the details of the replication etc. are not immediately relevant to the proof. That the replication is possible follows from condition 6.2.1, since every pair of Traces of \( G \) differing at a single FC place must have an appropriate termination. We can therefore ensure condition 6.4.1 in \( SN'(p) \) without any loss of parallelism.

Let step 3 in Alg. 5.3.1 be then replaced by the replication step described. Condition 6.4.1 actually ensures that \( SN'(p) \) is a top-down net with one or more 'input places' (i.e. places with one or more input transitions outside \( SN'(p) \)) in the same constituent psm which contains \( p \).

This completes the modifications needed in Alg. 5.3.1. The validity of steps 4-6 in Alg. 5.3.1 follows from arguments similar to those in Section 5.1, for type \#1 reductions. \( SN'(p) \) may not satisfy the output condition of transformation T3 of Section 5.1 (see \#(iii) above), but the reduction of \( SN'(p) \) to a single operator is still valid in view of condition 6.4.1.

B. Let \( G_c', G_r \) be as in Section 5.4, and consider the following condition on \( G_c' \):
condition  

\[ \forall R \in \mathcal{R}(G^c), \forall x \in P^{-1}(R), \quad [x \notin T^{-1}(G^c)] \Rightarrow x \in X^R_n \]

**Ex. 6.4.2** Consider the following net, where \( G^c = R \) is shown in bold lines:

![Net Diagram](image)

This net satisfies condition 6.2.1 but violates condition 6.4.2. As in the case of Ex. 6.4.1 above, replication of \( f_3, f_4, f_5 \) as shown below causes condition 6.4.2 to be satisfied:

![Replicated Net Diagram](image)

###

As in A above, we do not describe this replication in full since the details are not essential to the proof. As before, the replication is possible because condition 6.2.1 ensures that every pair of Traces of G differing at a single FC place has an appropriate termination.

Now consider the following modifications to Alg.5.4.1, which are possible only if condition 6.2.1 is satisfied.

#(i) Delete step 2 and sub-steps 3.1-3.4.

#(ii) Insert the following immediately after step 1:

step 1.5 ensure condition 6.4.2 through the replication step outlined above

#(iii) Modify step 1 exactly as Alg.5.3.1 is modified in A above.
In other words, replace SN(p) by SN'(p), found after deleting redundant constraints, apply T1,T2 but not T3 at sub-step 1.2, and omit sub-step 1.3 after ensuring condition 6.4.1 for SN'(p).

It can easily be seen that condition 6.4.2 ensures that G^C is a top-down net satisfying all the necessary conditions for a type#2 reduction. This enables us to delete sub-steps 3.1-3.4 from Alg.5.4.1. We thus avoid the need for separating G into G^C and G^R, and the subsequent linking together of the two at sub-step 3.4.

Note the similarities between the modifications carried out on Alg.5.3.1 and Alg.5.4.1. In each case we ensure that a top-down subnet is obtained in G (SN'(p) and G^C resp.) without the introduction of additional constraints into G, by means of the appropriate replication step.

This completes A and B. In C below we show that the above modifications constitute valid transformations.

C. In (i) and (ii) below, we prove the validity of the modifications described in A and B respectively. In other words, we show that the modified algorithms find a TDP simulating the original without introducing additional constraints into it.

(i) In Alg.5.3.1, the only points where there is a possibility of extra constraints being introduced are: transformation T2, sub-step 2.2 of T3, and sub-step 3.2. (Note that both T2 and T3 are called from Alg.5.3.1.) Sub-step 2.2 of T3 and sub-step 3.2 of Alg.5.3.1 involve an application of transformation A2 of Appendix II.

In the modified form the first of these does not involve any added constraints in G. This is so because we are now using
SN'(p) in place of SN(p), and therefore the set T_{last}(p) coincides with \(<p,r>\). We know that \(<p,r>\) satisfies part (i) of condition 6.2.1, i.e. \(\forall t \in \langle p, R \rangle, \forall t' \in \langle p, R \rangle', t \cap t' \neq \emptyset\). In other words, input and output transitions of \(<p, R>\) are pairwise mutually constrained. Therefore T2, diagrammed below, does not introduce any additional constraints in \(\overline{G}\).

Now consider sub-step 2.2 of T3 and sub-step 3.2 of Alg. 5.3.1. In the modified algorithm both these have been deleted (#(iii) and #(iv) in A) and therefore we have avoided the introduction of extra constraints into \(\overline{G}\).

We have thus accounted for every possible step in Alg. 5.3.1 which can cause a loss in the degree of parallelism in the given net \(\overline{G}\). The modifications are possible only because we can ensure condition 6.4.1 without adding any further constraints to \(\overline{G}\).

(ii) Now consider Alg. 5.4.1. In the modified version, step 1 causes no loss of parallelism because it has been modified exactly as Alg. 5.3.1 was modified in A above. Step 2 and substeps 3.2-3.3 in the original Alg. 5.4.1 are the only other points of introduction of additional constraints into \(G\), but condition 6.4.2 enables us to delete these steps (#(i) in B) and avoid this potential loss of parallelism. Thus each step
in Alg.5.4.1 where extra constraints can be introduced into $G$ has been accounted for. The basic strategy of Alg.5.4.1 has of course not changed in the modified version, since we still reduce $G^c$ through a type#2 reduction. The modifications to Alg.5.4.1 are possible only because we can ensure condition 6.4.2 without any loss of parallelism.

This completes part C, showing that if condition 6.2.1 is satisfied, then each of the algorithms of Chapter 5 can be suitably modified to yield a TDP equivalent to the original. We therefore have the following:

**Thm.6.4.1** Let $\mathbf{f} = (G, S_{op}, f_{op}, S_{pr}, f_{pr})$ be an FPP. If $G$ satisfies condition 6.2.1 then there exists a TDP equivalent to $\mathbf{f}$.  

The above proof of sufficiency was, necessarily, highly dependant on the algorithms of Chapter 5. Condition 6.2.1 enabled us to ensure conditions 6.4.1 and 6.4.2, respectively, of the nets $\overline{G}$ and $G^c$. These two conditions, in turn, permitted modifications (and in a certain sense simplifications) of Alg. 5.3.1 and Alg.5.4.1 s.t. the modified versions of the algorithms can be shown to involve no loss of parallelism (i.e. no introduction of additional constraints) in the nets $\overline{G}$ and $G$ respectively.

This brings to a conclusion the analysis of this Chapter, where we have determined necessary and sufficient conditions under which a parallel program has a top-down equivalent.
CHAPTER 7

CONCLUSIONS AND RELATED WORK

The major result of this paper is that of Chapter 6, i.e. the conditions under which a parallel program can be simulated by a top-down parallel program without any loss in the degree of parallelism attainable. In view of the fact that the conditions apply to interval reducible programs and structured programs as well when the definition of equivalence is appropriately modified (see Chapter 3), this paper constitutes a study of the implications of using methods of step-wise refinement for the design of parallel systems.

The advantages of using top-down design procedures for sequential programs are well-known. \(^{(15,17,18)}\) Ease in program validation, data-flow analysis, maintenance, modification, etc. are some of the well-known advantages of top-down design. For parallel programs, many of these problems become enormously more complex unless programs have the regular structure obtained through step-wise refinement using a restricted set of control schemes.

For parallel systems, therefore, we have discovered a performance factor that may potentially offset some of the advantages of top-down design: namely, a loss in the degree of parallelism attainable. Whether or not this loss of parallelism is significant for any specific problem will depend on other characteristics of the problem, such as the lengths of execution of the various operators. For a specific given problem, to increase the degree of parallelism achievable with
top-down design (at the cost of additional design effort), a module may be designed which is not any of the basic modules defined in Chapter 3. This module (to which there would be no top-down equivalent) may then be incorporated in the program which otherwise has been obtained through step-wise refinement. Whether such a module is designed will clearly depend on the probable execution times of the various operators, and the relative weight given to program speed in the over-all performance criterion.

The concept of 'maximally parallel' schemata has been analyzed in (13). In comparison with the results of that paper, we note that this thesis is concerned solely with finite state realizations of parallel program schemata, as represented by linear FCP nets. We have investigated the degree of parallelism realizable within this class (i.e. the class of FPPs) under the constraint of top-down design. In (14), Keller approaches the problem from a hardware point of view, looking at methods of realizing the potential parallelism in the central processor of a computer.

A natural theoretical question in this connection is: can any finite state realization of a commutative compact schema be represented as an FPP? The answer to this question will delineate the power of representation of FCP nets, and provide some valuable insight into the behaviour of Petri nets.

An active area of research into parallel systems concerns 'data-flow' systems. Some of the analysis of the type carried out in this paper can possibly be extended to these areas. (1,2) are broad surveys of the research in the area of parallel systems.
Appendix I

Proofs to Lemmas 1.4.3 - 1.4.6

Lemma 1.4.3 (proof)

Note: The proof is given here for components $S$, $S'$ etc. resulting from SM-reduction, but clearly a dual argument applies to components $R$, $R'$ etc. resulting from MG-reduction.

Let $S'$ be as given and let $AS'$ be the corresponding SM-allocation. We modify $AS'$ to yield an allocation $AS$ as follows:

$$
\forall t \in T, AS(t) = \begin{cases} 
AS'(t) & \text{if } t \text{ is on } S' \text{ or if } AS'(t) \text{ is on an elementary path } \tau(x,t) \text{ from a vertex } x \text{ on } S' \text{ to } t; \\
p & \text{otherwise, where } p \in t \text{ is on an elementary path } \tau(x,t) \text{ from a vertex } x \text{ on } S' \text{ to } t.
\end{cases}
$$

We now show that SM-reduction corresponding to $AS$ yields exactly the SCSM-component $S$ described in the statement of the Lemma.

Assume the opposite, i.e. let $\bar{S}$ be another SCSM-component disjoint with $S'$ obtained through this SM-reduction. Then there is a cycle $\gamma$ in $\bar{S}$ s.t. $\forall t \in T$ on $\gamma$, $AS(t)$ is on $\gamma$. But this contradicts the above definition of $AS$. Our assumption is, therefore, wrong, and $S$ is the only component resulting on SM-reduction with $AS$. Finally, that $S'$ is contained in $S$ follows at once from the definition of $AS$, completing the proof.

Lemma 1.4.4 (proof)

Assume the opposite. Then the paths shown below must exist in $G$; let $R'$ denote the subgraph formed by these paths.

![Diagram](image-url)
Now consider an MG-allocation AM on G which satisfies the condition: \( \forall p \in P, p \) is on \( R' \) \( \implies AM(p) \) is on \( R' \). Clearly then \( R' \) will be contained in a maximal strongly connected component obtained on MG-reduction with AM. From Lemma 1.4.3 we can therefore obtain an SCMG-component \( R \) of G s.t. \( R' \) is a subgraph of \( R \). Since G is a well-formed net, however, we know that \( R \) is a marked graph. This contradicts the fact that the input degree of \( y \) in \( R' \) is greater than 1. Our assumption must be wrong, and the result follows.

**Lemma 1.4.5 (proof)**

Let \( AM' \) be the MG-allocation yielding strongly connected component \( G' \) on MG-reduction at step (ii). Modify \( AM' \) to yield MG-allocation AM as follows:

\[
\forall p \in FC(G), \ AM(p) = \begin{cases} 
AM'(p), & \text{if } p \text{ is on } G' \text{ or } AM'(p) \text{ is on an elementary path } \tau(p,x) \text{ from } p \text{ to a vertex } x \text{ on } G'; \\
t, & \text{otherwise, where } t \in p' \text{ is on an elementary path } \tau(p,x) \text{ from } p \text{ to a vertex } x \text{ on } G'.
\end{cases}
\]

Then \( G' \) must be contained in the only SCMG-component resulting on MG-reduction with AM (as in Lemma 1.4.3) and the result then follows.

**Lemma 1.4.6 (proof)**

Assume the opposite. Then the following paths must exist in G. Let \( R' \) be the subgraph of G formed by these paths. Note that we have shown cycle \( \tau \) to consist of i-disjoint paths \( \tau_1(x,y) \) and
\( \pi_2(y, x) \).

Consider an SM-allocation AS on \( G \) s.t. \( \pi \) is in the resulting SCSM-component \( S \). Transition \( t \in x' \) must therefore be on \( S \). It follows that an elementary path \( \alpha(t', p') \) exists in \( S \) s.t. \( t' \) is on \( \tau \), \( p' \) is on \( \pi \), and no other vertex of \( \alpha \) is on \( \tau \) or \( \pi \). So the following paths must exist in \( G \), where \( t' \) may in fact coincide with \( t \).

From the premise in the statement of the Lemma, it follows that \( p' \) is on \( \pi_2 \), and not on \( \pi_1 \). But then the pair of i-disjoint paths from \( t' \) to \( p' \) violates Lemma 1.4.4. Our assumption must be wrong and the result follows.
Appendix II

A Transformation on Proper Marked Graphs

Recall the definition (Chapter 3) of Proper Marked Graphs. A PMG is simply a live and safe marked graph modified so that a unique initial transition and a unique terminal transition can be identified. We now describe a way of rearranging the vertices of a PMG so as to eliminate the occurrences of pipeline branches from the graph. The transformation described here, denoted as transformation A2, and shown to be a valid transformation, is used in Algs. 5.3.1 & 5.4.1 of Chapter 5, and in transformation T3 of Section 4.2.

Let \( G_m = (T_m, P_m, E_m) \) be a PMG, shown schematically below:

![Diagram of a PMG with initial and terminal transitions]

**Def. A2.1** A place \( p \in P_m \) is a redundant constraint in \( G_m \) if the following conditions are satisfied, where \( \{t_1\} = p \) and \( \{t_2\} = p' \):

i. \( t_1 \neq \{p\} \) and \( t_2 \neq \{p_2\} \),
ii. there is a blank path \( \Pi \) in \( G_m \) from \( t_1 \) to \( t_2 \) s.t. no \( x \in t_1 \cap t_2 \) is on \( \Pi \).

[Note that this definition corresponds exactly with the latter definition (in Chapter 6) of redundant constraints in a single Trace of a live and safe FCP net.]

Then transformation A2 consists of an appropriate sequence from the three sub-transformations A2.1 - A2.3 described below.

**A2.1** Let \( p \) be a redundant constraint in \( G_m \). Delete \( p \) from \( G_m \),
along with its input and output edges.

A2.2 Let \( \{t_1, t_2, t_3\} \subseteq T_m \) be in the following configuration in \( G_m \), where \( t_3 \) satisfies \( t_3=\{p_2\} \):

Then replace the above subgraph of \( G_m \) by the following:

A2.3 Let \( \{t_1, t_2, t_3\} \subseteq T_m \) be in the following configuration in \( G_m \), where \( t_2 \) satisfies \( t_2=\{p_1, p_2\} \):

Then replace the above subgraph of \( G_m \) by the following:

That A2.1 is a valid transformation follows at once from well-known properties of marked graphs\(^4\). That A2.2 and A2.3 are valid transformations follows from similar arguments, with
the difference that they involve introduction of additional constraints into $G_m$. [The loss of parallelism involved in A2.2 and A2.3, in the form of additional constraints introduced, is unavoidable under certain conditions when we attempt to simulate a parallel program by a top-down parallel program. This question is analyzed thoroughly in Chapter 6.]

**Ex.A2.1** Consider the PMG shown below. In bold lines in the diagram we have shown the pipe-line branches of the graph.

We now illustrate an application of A2 on the above graph to yield a marked graph simulating it which has no pipe-line branches. In other words, we now show how an appropriate application of A2 eliminates the pipe-line branches from a PMG.

We first apply A2.3 to transitions $t_5, t_7, t_9$ to yield the following net:

Now an application of A2.2 to $t_9, t_{10}, t_8$ followed by an application of A2.1 to $q$ yields:
Note that the pipe-line branches have now been absorbed into other transition-sequences of the marked graph. It can easily be verified that the above graph simulates the original.

Finally, note that the above transformation would work equally well on a Trace of a linear FCP net. From the analysis of Section 4.3 we know that a Trace R of G which does not contain 'start' has the form:

\[
\begin{array}{c}
\text{t}_2 \\
\text{o} \\
\text{t}_1
\end{array}
\]

where \( x \in X^R_e \). Therefore \( t_1 \) and \( t_2 \) may be viewed as the initial and terminal transitions of a PMG. We also know, of course, that a Trace of G containing 'start' has the form of a PMG.
Appendix III

A Transformation on $SN^R(x)$

Recall that $SN^R(x)$ is the subnet in the neighbourhood of $x$ in the net $G^R$, where $x$ is an exit of $G^C$, i.e. $x \notin T^{-1}(G^C)$, as defined in Section 5.4. The transformation we describe here is carried out on each exit of $G^C$ at step 3.3 of Alg.5.4.1.

It can easily be shown, using arguments of Lemmas 4.2.1 & 4.2.2 that $T_{\text{last}}(x) \subseteq T^R_{\text{last}}(x)$. Now consider the following net.

Ex.A3.1

In the above linear FCP net let $R = G^C$ be as shown in bold lines, and let $x_1, x_2$ be as shown. Then $T^R_{\text{last}}(x_1) = \{ t_1, t_2 \}$, while $T_{\text{last}}(x_1) = T_{\text{last}}(x_2) = T^R_{\text{last}}(x_2) = \{ t_2 \}$.

The aim of the transformation described here is to ensure that $T^R_{\text{last}}(x) = T_{\text{last}}(x)$ for every exit $x$ of $G^C$. We first illustrate the transformation on the subnet $SN^R(x_1)$ of the above net, which is drawn separately below.

The above subgraph will be transformed to the following, when the transformation (denoted here A3) is applied to it.
A3 then consists of the following steps:

**step 0** Choose arbitrarily a transition $t \in T_{\text{last}}(x)$, and introduce at its input the $\lambda$-transition as shown below.

$$
\begin{array}{c}
\text{Original Transition} \\
\text{New Transition}
\end{array}
$$

**step 1** Delete the output edges of the places in the set
$$
\left[ T_{\text{last}}^R(x) - T_{\text{last}}(x) \right] \cap P^{-1}[SN^R(x)].
$$

**step 2** Add an edge from each of the places of step 1 above to the $\lambda$-transition introduced at step 0. Remove the tokens, if any, from these places.

It can easily be seen that the example subgraph $SN^R(x_1)$ was transformed in a manner consistent with the above procedure. That this transformation on $G^R$ is valid follows from arguments similar to those of Lemmas 4.2.1, 4.2.2 etc., on considering a SCSM-component of $G^R$ and showing that it is transformed into a valid SCSM-component of the output net.

We complete this description of A3 with the top-down net obtained from the net of Ex.A3.1 when Alg.5.4.1 is applied to it. The transformed subnet $SN^R(x_1)$ can be seen in bold lines, intuitively indicating the need for A3 in Alg.5.4.1 at step 3.3. (p.t.o.)
In the above diagram, superscripts 1 and 2, respectively, distinguish vertices in $G^r$ and $G^c$ which have the same associated operator or predicate. Transitions $t_6^2, t_7^1$ and $t_8^2$ are introduced at step 3.4 after $G^c$ has been transformed separately at steps 3.1, 3.2 of Alg. 5.4.1.
References


Additional bibliography (not referenced)


